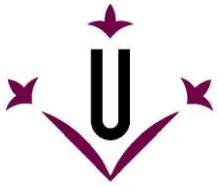


Universitat de Lleida

Escola Politècnica Superior

Enginyeria Tècnica en Informàtica de Sistemes

TREBALL FINAL DE CARRERA

Aplicatiu web per la gestió dels continguts del restaurant La Paradeta mitjançant Symfony2

Autor: **Jorge Gómez Arriba**

Director: **Carles Mateu Piñol**

4 de setembre de 2012

Índex de continguts:

Índex de figures:

Índex de taules:

1 Introducció

1.1 Motivació

L'elecció del projecte a realitzar, no va ser una tasca gens amena. Vaig realitzar diverses entrevistes amb els responsables dels diferents departaments amb la finalitat d'esbrinar si tenien en ment la realització d'algun treball, però tots van respondre'm que penses un tema i els hi proposes. Finalment, vaig decidir-me per la programació web, com a resultat que durant la titulació vaig realitzar diferents tallers dedicats a aquesta, a més, el bloc optatiu que vaig elegir va ser precisament el d'Internet, orientat a aplicacions, administració i seguretat, tres temes fonamentals en la realització del present projecte final de carrera.

Un cop decidit el tema, havia de trobar alguna proposta de temàtica de creació web. Un conegut acabava d'obrir un restaurant i necessitava crear-se una pàgina web per tal de publicitar-se i donar-se a conèixer, amb el motiu que aquest està ubicat en un poble de muntanya força turístic, i per tal de millorar el rendiment del seu negoci, era essencial la realització d'aquesta.

El que més vaig valorar a l'hora d'elegir projecte, va ser la motivació de realitzar un treball innovador, el qual impliqués l'estudi de noves tecnologies; hem donés la possibilitat d'una participació continuada i poder ampliar els coneixements assolits durant la titulació.

L'aplicació desenvolupada és de caire comercial, on la seva finalitat és la difusió dels continguts del restaurant. Sota aquesta premissa s'ha desenvolupat tota la lògica d'aquesta. Com a plataforma tecnològica, s'ha optat per emprar el framework Symfony 2, un dels més complerts del panorama actual.

Finalment, m'agradaria donar els pertinents agraïments a les següents persones les quals han format part de la confecció del present projecte final de carrera:

- Als meus pares i demés familiars, pel suport que m'han mostrat durant tot el procés que ha comportat l'elaboració del projecte.
- A Carles Mateu, tutor del projecte, el qual m'ha guiat durant tot el procés d'elaboració d'aquest.
- A Javier Eguiluz, programador de Symfony 2, el qual molt pacientment m'ha respost a tot tipus de qüestions formulades sobre la lògica de funcionament del framework.

1.2 Introducció general

Des de l'aparició de les primeres pàgines web accessibles gràcies a l'expansió i reconeixements de l' HTML, els continguts disponibles s'han fet bàsics per la societat actual i aquestos han anant oferint-nos noves capacitats en la navegació i la connectivitat a través de la xarxa.

En un principi, la web estava orientada a oferir continguts estàtics. L'autor publicava continguts i esperava que els usuaris poguessin accedir a aquestos i consultar-los; per part dels usuaris, esperaven continguts disponibles i sense canvis que fossin accessibles en qualsevol moment.

Però des de llavors, les aplicacions disponibles a la xarxa han evolucionat notablement. Els paradigmes en que es basava la web s'han voltejat, fent ús de les característiques bàsiques que es van assentar en els seus inicis enfocats a continguts estàtics, es va passar a oferir continguts dinàmics, sense haver de realitzar una costosa i sovint complicada navegació. S'ha produït un canvi radical en quant a la presentació i la interacció amb l'usuari, el qual ja no desitja una plana on pugui consular informació, sinó una on pugui interactuar amb aquesta i consultar els seus continguts d'una manera fàcil i agradable, possibilitant fer-los actors principals dels continguts i funcionalitats que es presenten.

Un altre aspecte que ha possibilitat aquesta transició, és l'eliminació de les barreres de connectivitat que podien haver entre les aplicacions i la xarxa. No és que s'hagi produït una renovació en l'estructura i tecnologia emprades per la xarxa, sinó que ha estat fruit d'un procés d'evolució gradual, en la qual aquesta s'ha anant adaptant a les necessitats requerides pels usuaris.

El present projecte és un clar exemple dels paràgrafs anteriors. L'usuari pot adquirir (interactuar) els articles (continguts) d'una forma fàcil i simple, o geolocalitzar el restaurant sense la necessitat d'emprar altres tipus de software, on la navegació passa a ser un procés d'estètica i agradable.

1.3 Descripció del projecte

La temàtica del projecte a desenvolupar és l'elaboració d'una aplicació web amb serveis orientats al comerç electrònic i la gestió d'aquestos, fent ús del framework Symfony 2 pel desenvolupament d'aquesta.

L'aplicació desenvolupada és de caire comercial, on els serveis presentats tenen com a finalitat la difusió dels continguts presentats pel restaurant. L'especificació descrita pel gerent d'aquest, es basa en la difusió dels articles alimentaris, la geolocalització del

restaurant, i l'aspecte més important, l'elaboració d'una interfície la qual atregui al major públic possible.

Sota aquestes premisses es va realitzar un primer esbós del disseny tècnic e interfície a realitzar.

1.4 Objectius

Els principals objectius que es buscava assolir amb la realització d'aquets projecte són:

- ✓ Masterització del framework Symfony 2 pel desenvolupament web.
- ✓ Comprendre la lògica de funcionament dels serveis de l'aplicació web.
- ✓ Dinamització dels recursos.
- ✓ Implementació d'una política de permisos d'accés pels usuaris.
- ✓ Implementació d'un gestor de reserves fàcil d'emprar i de consultar.
- ✓ Implementació d'un entorn per a la gestió d'articles, usuaris i reserves.
- ✓ Implementació d'un sistema de geolocalització del restaurant basat en l'api v3 de Google Maps.
- ✓ Implementació d'un control de reserves i gestió d'usuaris.
- ✓ Implementació d'una estructura de dades i gestió d'aquesta que permeti afegir nous continguts i modificar-los sense la necessitat de realitzar tasques d'adaptació.
- ✓ Implementació d'una estructura de dades i de gestió que permeti diferenciar els serveis en diferents subtipus.
- ✓ Adaptació d'un estil i una identitat pròpia per l'aplicació.
- ✓ Correcció dels possibles *bugs* existents.

1.5 Requisits de l'arquitectura

Els requisits necessaris per tal de poder desenvolupar l'aplicació són els següents:

- ✓ Ordinador portàtil Packard Bell EasyNote amb processador dual core.
- ➢ Sistema operatiu Ubuntu Lucid versió 10.4

- Servidor Apache on poder hospedar Symfony2, amb accés als logs del compte i suport per PHP 5, la versió 4 no proporciona orientació a objectes, cosa que es indispensable per l'execució del framework.
- Gestor MySQL versió 5 per suportar procediments d'emmagatzematge i l'accés a les diverses taules desenvolupades al llarg d'aquest.
- Versió del framework Symfony 2.10, per tal de suportar la programació sobre twigs.

Posteriorment, es farà us dels següents recursos per tal de fer accessible l'aplicació web a tots els usuaris de la xarxa.

- Domini web per proporcionar accés a l'aplicació a través de la xarxa, el qual s'ha adquirit de la pàgina http://www.ovh.es/dominios/; l'aplicació desenvolupada s'ha registrat al domini anterior i és accessible mitjançant la següent URL http://www.restaurantelaparadeta.com
- Servei de hosting per tal d'hospedar els arxius necessaris perquè l'aplicació funcioni correctament (base de dades, imatges d'articles, etc.), el qual s'ha adquirit de la pàgina http://www.ovh.es/hosting/; dona suport per a tecnologia Apache, MySQL i PHP.

Com la finalitat de l'aplicació és comercial, els requisits per tal de poder accedir a aquesta, seran bàsics. Qualsevol usuari que tingui accés a la xarxa, ha de tenir accés a l'aplicació, sense la necessitat d'instal·lació de software extern.

Un dels requisits especificats pel client, era que la navegació per l'aplicació havia de ser un procés agradable, molt fàcil d'emprar i, sobretot, atractiu a la vista. No s'havia d'elaborar una aplicació per a usuaris de nivell intermedi o avançat, sinó per a usuaris novells en la gran immensitat que és la xarxa. Tot això va condicionar l'elaboració del disseny tècnic de l'aplicació, on vaig haver d'elaborar diverses interfícies sobre les quals hauria de mostrar-se l'aplicació.

1.6 Filosofia d'un framework

Un framework és una estructura conceptual i tecnològica de suport definit, normalment amb mòduls de software concrets, amb base que altres projectes de software poden ser desenvolupats i organitzats més fàcilment, optimitzar el treball requerit pel programador. Típicament, inclou suport de programes, biblioteques i un llenguatge propi, per tal de desenvolupar i unir les diferents parts que conformen l'aplicació.

Representa una arquitectura de software que modela les relacions generals de les enti-

tats del domini, i proveeix d'una estructura i una especial metodologia de treball, la qual empra les aplicacions del domini.

Aquestos són dissenyats amb la intenció de facilitar el desenvolupament del software, permetent als dissenyadors i programadors dedicar bona part del temps a identificar els requeriments d'aquest. Per exemple, un programador que empra un framework per desenvolupar un portal web d'un comerç, pot dedicar-se a implementar com un usuari pot realitzar les compres, en lloc de preocupar-se de com es controla l'accés o la navegació entre les pàgines en una forma lliure d'errors. No obstant això, l'ús d'un framework implica, sovint, la manipulació de codi innecessari, i sobretot, la preponderància de frameworks competitius i complerts, significa que el temps que abans s'emprava programant i dissenyant, ara cal emprar-lo en entendre el funcionament intern d'aquestos i la seva lògica de negoci.

Fora de les aplicacions informàtiques, aquestos poden considerar-se com el conjunt de processos tecnològics emprats per resoldre un problema complex. Es l'esquelet sobre el qual varis objectes són integrats per facilitar una solució donada.

1.6.1 Arquitectura d'un framework

L'arquitectura d'aquestos es basa en el model MVC (Controlador => Model => Vista), a causa que fragmenta la nostra programació. Separa les dades de l'aplicació, la interfície d'usuari i la lògica de negoci en tres components diferents. Es una base de programació que encara les seves necessitats als seus descendents (manipulant-los d'una forma estructural i/o en cascada), possibilitant qualsevol resposta davant les necessitats dels seus membres o seccions. En una aplicació web, el model MVC ha de ser capaç de:

- Controlador: Respondre a les acciones de l'usuari, invocació de peticions al model i proveir l'encaminament d'arxius, classes, mètodes i funcions, a més de proveir l'aplicació d'operacions lògiques.
- Model: Representar la informació amb la qual el sistema opera, la qual es troba emmagatzemada en una base dades.
- Vista: A aquest element li correspon presentar el model en un format adequat per interactuar, també anomenada interfície d'usuari.

A la secció 3.1.2, s'amplia la informació de l'arquitectura MVC introduïda en aquest apartat.

1.6.2 Disseny d'un framework

La principal crítica que es sol realitzar als frameworks és que son massa intrusius, a causa que obliguen al programador a treballar d'una determinada forma. L'inconvenient més important és la resposta a les peticions d'usuari, a conseqüència que aquestos requereixen al programador interposar accions entre la petició i la resposta de l'aplicació.

Symfony 2, en canvi, ha estat dissenyat per evitar aquest inconvenient i per interferir el mínim possible en el *corrent natural* de les peticions HTTP.

Si un portal web està composat per pàgines estàtiques, el seu funcionament és el següent:

- L'**usuari** sol·licita una pàgina mitjançant la seva URL.
- El **servidor web** retorna el contingut de la pàgina que correspon a la URL.

En el cas dels portals web dinàmics controlats per una aplicació, el seu funcionament és el següent:

- L'**usuari** sol·licita una pàgina mitjançant la seva URL.
- L'**aplicació** genera el contingut corresponent a la URL, on el programador és l'encarregat de donar resposta a totes les peticions generades.
- El **servidor web** retorna el contingut que li passa l'aplicació.

En canvi, si l'aplicació està programada mitjançant Symfony 2:

- L'**usuari** sol·licita una pàgina mitjançant la seva URL.
- **Symfony 2** transforma la petició en un objecte de tipus *Request* i li retorna a l'aplicació.
- L'**aplicació** genera el contingut corresponent a la petició i retorna a Symfony2 la resposta mitjançant un objecte de tipus *Response*.
- **Symfony 2** transforma l'objecte *Response* en contingut llegible pel servidor web.
- El **servidor web** retorna el contingut que li passa Symfony 2.

Symfony 2 s'adapta perfectament al model petició + resposta HTTP. La seva arquitectura s'ha dissenyat per facilitar la creació d'un objecte de tipus *Response* a partir d'un objecte de tipus *Request*. Fora d'aquest àmbit, Symfony 2 desapareix i deixa que sigui el programador el que realitzi la implementació al seu gust.

A la secció 3.1.12 i 3.1.13, s'amplia la informació sobre els objecte de tipus *Request* i *Response* respectivament.

1.7 Procés tecnològic

Un procés tecnològic el podem definir com el camí que s'ha de recórrer des de que es requereix un problema (en aquest cas la realització d'una aplicació web) fins que generem un objecte que el soluciona. Aquest es composa de tres fases:

- **Fase de desenvolupament:** El programador ha de realitzar un estudi previ, on s'identificarà el problema, planificarà els recursos necessaris, els requeriments d'aquest i proporcionarà al client una possible solució.
- **Fase d'execució:** El programador redactarà un disseny tècnic, el qual inclourà una descripció detallada de la resolució del problema, i realitzarà la implementació pertinent. El client rebrà el producte final de caràcter experimental i determinarà un veredicte.
- **Fase de producció:** El programador realitzarà un seguit de proves unitàries per tal d'arribar a determinar si l'objecte generat satisfà les necessitats de l'usuari. Durant el transcurs d'aquesta fase, el programador ha de realitzar una avaluació, en qualitat d'emfatitzar les possibles deficiències de la implementació realitzada, i assenyalar els mètodes per millorar-les.

En cas que l'objecte final no superi amb èxit la fase de producció, es reenviarà a la fase d'execució, on es repetirà tot el procés a partir del punt assenyalat.

1.8 Memòria

En aquest document es presenta informació detallada sobre la realització d'un projecte final de carrera d'enginyeria tècnica en informàtica de sistemes. Els continguts presentats en aquesta ens descriuen els motius, objectius, disseny tècnic i proves unitàries sobre les quals estan constituïdes totes les parts de l'aplicació, donant una especial importància a les decisions preses durant el desenvolupament del disseny tècnic.

El primer capítol presentat en aquesta és una breu introducció on s'exposen els motius pels quals he desenvolupat aquesta aplicació, els requisits de l'arquitectura i una breu menció al disseny d'aquesta.

El segon capítol ens presenta dos punts clarament diferenciats però basics en la realit-

zació del projecte. El primer ens descriu la modelització i el procés que s'ha recorregut durant el període de desenvolupament. El segon és una simulació dels costos que suposaria l'elaboració d'aquest en un àmbit empresarial, fent us d'un anàlisis econòmic.

El tercer capítol busca familiaritzar-nos amb la terminologia emprada durant la realització, on s'esmenta algunes de les claus essencials per tal de començar-lo a comprendre'l conceptualment.

El quart capítol ens presenta el disseny tècnic elaborat durant la realització del projecte. Concretament, fa una menció especial a la implementació que s'ha seguit durant el període de desenvolupament.

El cinquè capítol ens presenta tot el seguit de proves unitàries a les que s'ha sotmet l'aplicació durant la fase de producció, la finalitat de la qual és corregir els possibles *bugs* existents.

El sisè capítol ens presenta tot el seguit de conclusions i treballs futurs extrets de l'elaboració de l'aplicació.

El setè capítol ens presenta tota la bibliografia i netgrafia consultada.

El vuitè capítol ens presenta l'annex on es detalla la metodologia d'instal·lació i configuració de Symofny2 i de la base de dades.

2 El projecte: Aplicació la Paradeta

2.1 Estat de l'art de l'aplicació

Durant la fase de desenvolupament, es van realitzar un seguit de reunions amb el gerent del restaurant, per tal d'aclarir els continguts i les funcionalitats que s'havien de presentar en l'aplicació. El client va demanar que es realitzes una interfície molt fàcil d'emprar i sobretot molt estètica, sobre la qual s'hauria de modelitzar la informació. A continuació es presenta el disseny de la interfície desitjada pel client:

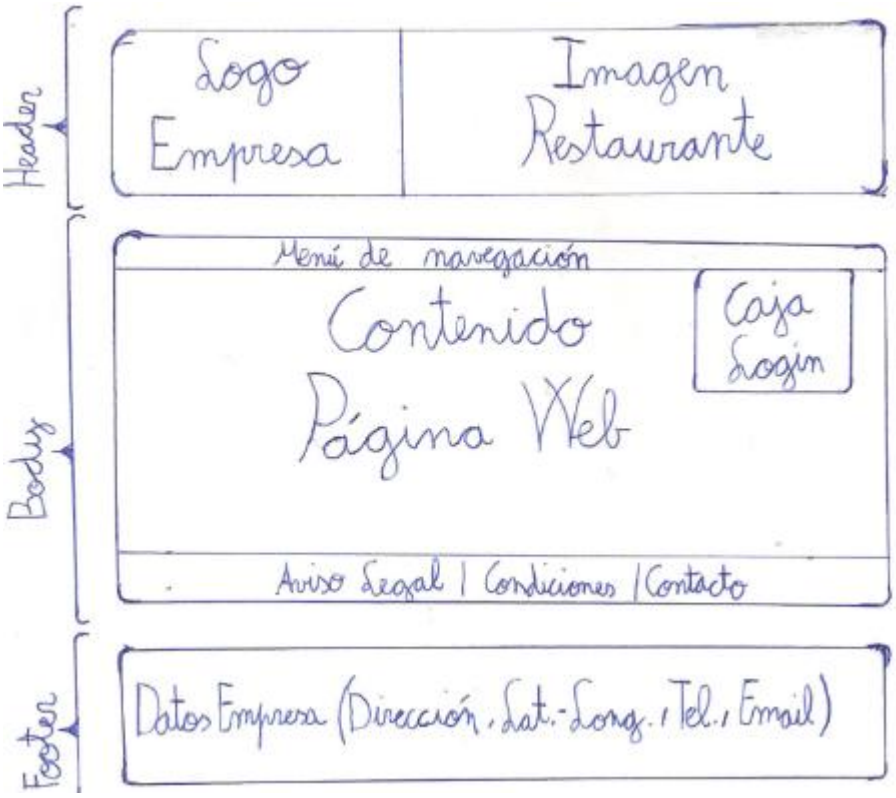


Figura 1: Especificació de la presentació de l'aplicació

Les funcionalitats que l'aplicació hauria de satisfer són les següents:

- ✓ Mostreig de la portada.
- ✓ Mostreig de sidres, vins i caves.
- ✓ Mostreig de menús disponibles i configurables per l'usuari.
- ✓ Mostreig del menú especial disponible els caps de setmana.
- ✓ Mostreig de les tapes disponibles.

- ✓ Implementació d'un formulari de reserves.
- ✓ Implementació d'un sistema de geolocalització.
- ✓ Mostreig de condicions de compra i avís legal.
- ✓ Implementació d'un formulari de contacte.
- ✓ Implementació d'un sistema de *login* d'usuaris.
- ✓ Implementació d'un sistema de mostreig de compres, del perfil i de reserves realitzades per part de l'usuari.
- ✓ Implementació d'un sistema de gestió de continguts (productes, usuaris, reserves, compres).

Al capítol 4 s'amplia la informació introduïda en aquesta secció.

L'aplicació desenvolupada, consta de dues parts:

- ➢ Frontend: Es la part pública de l'aplicació, la qual es subdivideix en dos segments clarament diferenciats, seguint la lògica de les versions anteriors del framework:
  - Pública: Aquesta pot esser accedida per usuaris no registrats. La finalitat d'aquesta és publicitària, on l'usuari tant sols podrà consultar els menús disponibles, realitzar reserves i emprar el sistema de localització.
  - Privada: Aquesta és d'accés restringit a l'usuari de l'aplicació, la qual sol pot esser accedida per usuaris acreditats. Aquestos podran realitzar i visualitzar compres, reserves i modificar el seu perfil.
- ➢ Backend: És la zona d'administració, la qual sol podràs ser accedida pel gerent. Aquesta permetrà visualitzar, modificar i afegir productes, usuaris, compres i reserves.

Un cop presentades les funcionalitats comuns de l'aplicació, s'havia d'especificar els *wireframes* (estructuració dels continguts específics de les principals pàgines de l'aplicació).

2.1.1 La portada

La portada és l'element més important de l'aplicació, a causa que és la primera impressió que l'usuari rebrà quan la visiti. Segons l'especificació del client, aquesta

havia de contenir un text de presentació on es mostressin d'una forma clara i concisa les principals característiques del restaurant. La premissa sobre la qual s'ha desenvolupat l'aplicació es que una bona estructuració garanteix un rendiment òptim.

Per tal de millorar la seva estètica, es van introduir diverses fotografies referents a l'interior del restaurant, integrades al text de presentació. A continuació es mostra la distribució especificada pel gerent:

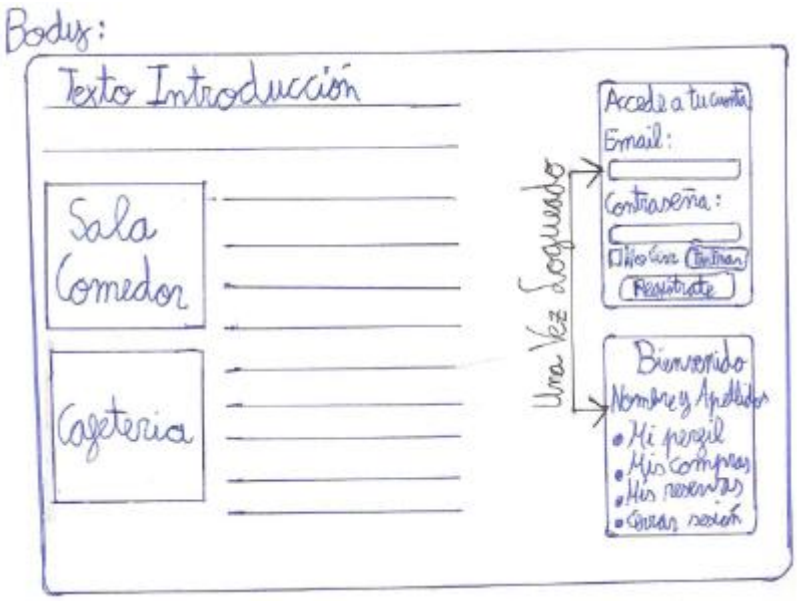


Figura 2: Especificació de la portada

2.1.2 Mostreig i compra d'articles

Les planes desenvolupades per modelitzar el catàleg d'articles segueixen el mateix patró. S'introdueix una breu descripció introductòria la qual engloba tots els articles de la mateixa classe (Vins & Sidra, Menú, tapes) i seguidament, es mostrarà el catàleg d'articles disponibles distribuïts de la següent forma:

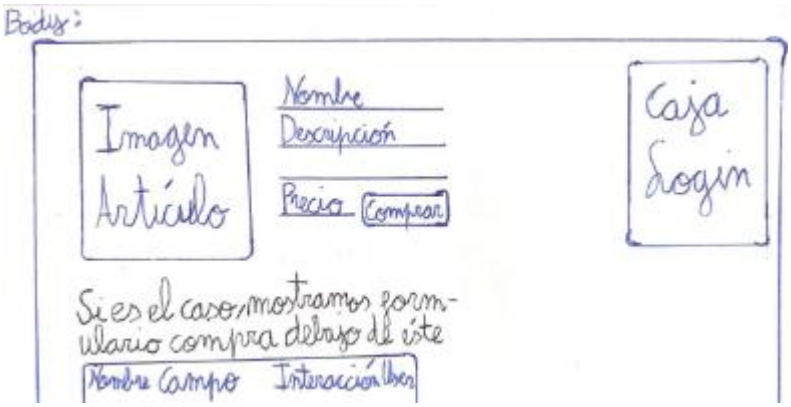


Figura 3: Especificació del mostreig d'articles

La descripció a mostrar dels diferents articles és la següent:

- ✓ Nom del producte.
- ✓ Breu descripció d'aquest.
- ✓ Preu.
- ✓ Fotografia.
- ✓ Total d'articles, si escau.
- ✓ Condicions especifiques de compra, si escau.

La peculiaritat que assoleix el disseny realitzat, és que l'usuari pot configurar el seu propi menú, donant tota a llibertat a la seva presa decisions, fent-los actors principals de l'aplicació. Això comporta que l'usuari pugui combinar al seu gust tots els productes disponibles a l'aplicació.

2.1.3 Sistema de localització

Un dels atractius principals de l'aplicació especificats pel gerent, era l'elaboració d'un sistema de creació de rutes mitjançant l'api de Google Maps. Això es pot traduir en la implementació d'un cercador de rutes des de la posició actual del client fins a la ubicació del restaurant, emprant les seves coordenades, on es mostrés la ruta que hauria de recórrer per tal d'arribar-hi.

Per l'elaboració de la present pàgina, es va seguir la mateixa metodologia fins ara, amb la peculiaritat que el text introductori interactua amb una imatge de la part exterior del restaurant. A continuació es mostra la distribució:

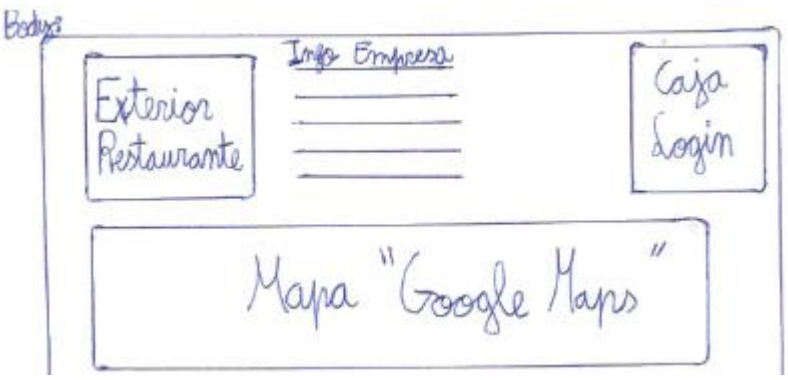


Figura 4: Especificació del sistema de localització

2.1.4 Pàgines estàtiques

Les pagines estàtiques són aquelles on es mostra les condicions de compra, avís legal i el formulari de contacte,invariables en el temps. El seu disseny seguirà l'estàndard de les pagines disponibles a la xarxa.

2.1.5 Formularis de registre, reserves i compres

Els formularis desenvolupats, presenten la informació que l'usuari ha de proporcionar a l'aplicació per tal que aquest pugui interactuar amb ella.

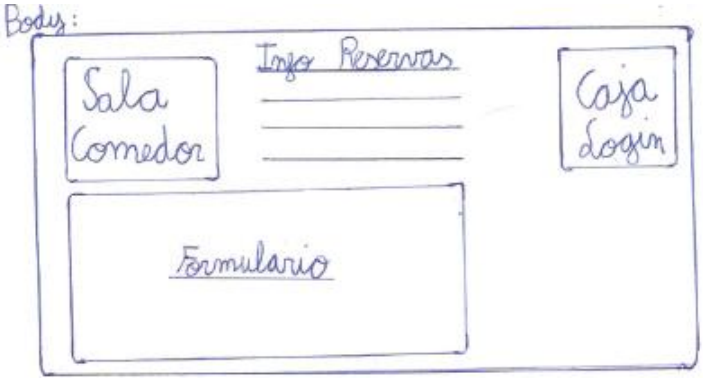


Figura 5: Especificació de la presentació de reserva

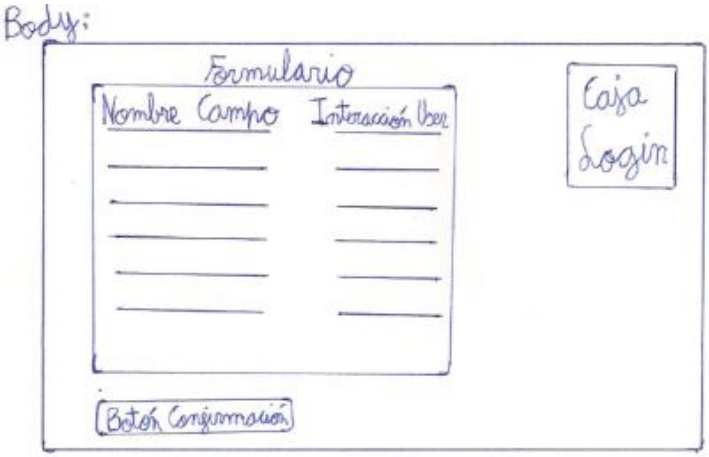


Figura 6: Especificació del formulari

Les dades que l'aplicació requereix, si escau, es la següent:

- ✓ Correu electrònic.
- ✓ Nom.
- ✓ Cognoms.
- ✓ Contrasenya per tal d'accedir a l'aplicació.
- ✓ Data de naixement.
- ✓ Direcció.
- ✓ Telèfon de contacte.
- ✓ DNI.
- ✓ Comentaris.
- ✓ Comensals.

Tots els camps presentats anteriorment, són validats per tal de comprovar que l'usuari realitzi una correcta utilització de l'aplicació.

2.1.6 Sistema de Login

El sistema de *login* desenvolupat en l'aplicació consta de tres parts:

- ➢ Frontend:
  - Una pública on poden accedir tots els usuaris de l'aplicació sense la necessitat d'autenticar-se. Aquesta consta de totes les parts esmentades anteriorment.
  - Una d'accés restringit, la qual sol podrà esser accedida per usuaris acreditats. Aquesta estarà formada per la realització i visualització de compres, de reserves i del perfil de l'usuari. El sistema per tal de restringir l'accés, es composa d'un discriminador en la URL de la forma *URL_aplicació/**usuaris**/accions_usuari*, on tot el que comporti emprar el discriminador *usuari*, pertanyerà a la part privada. Per tal de fer accessible el *login* i el registre a usuaris no *loguejats*, el discriminador */usuaris/registre* i el */usuaris/login*, pertanyeran a la part pública.
- ➢ Backend:

- Una d'accés restringint a tots els usuaris, no *loguejats* i *loguejats*, a la qual tant sols tindrà accés l'administrador (el gerent) de l'aplicació. Aquesta seguirà la mateixa lògica que l'anterior i emprarà el discriminador *backend*, on la URL es composarà de *URL_aplicació****/backend****/accions_usuari*. Les funcionalitats que aquesta ha de respondre són les següents:
  - Modificació, creació i eliminació de productes.
  - Mostreig de compres.
  - Mostreig de reserves.
  - Mostreig d'usuaris.

2.1.7 Perfil de l'usuari

Aquest, permetrà a l'usuari visualitzar i modificar les seves dades, que no formin part explicita del funcionament de l'aplicació (salt, data d'alta a l'aplicació), en una composició estructurada com un formulari.

2.1.8 Mostreig de compres i reserves

Aquest, permetrà a l'usuari (frontend) visualitzar les seves compres i les seves reserves en una composició estructurada en forma de taula.

Figura 7: Especificació de la presentació de compres i reserves

La part d'administració (backend), permetrà consultar totes les dades dels usuaris, mo-

dificar, afegir i eliminar els productes, i consultar les reserves i compres en una composició estructurada en forma de taula.

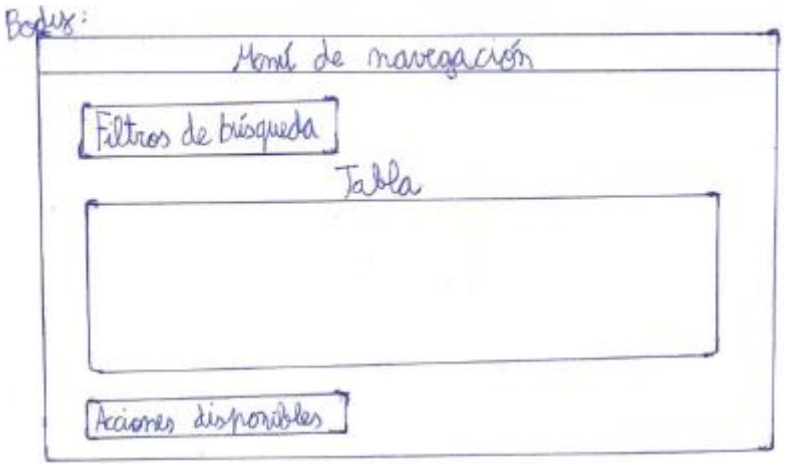


Figura 8: Especificació de la presentació d'administrador

2.2 Planificació i anàlisis econòmic

2.2.1 Planificació

El primer a confeccionar abans de la realització del disseny de l'aplicació, és la planificació de recursos i temporització d'aquestos. La planificació inicial realitzada juntament amb l'estructuració del projecte, dista molt de la final.

Inicialment, es va preveure una temporització d'aproximadament 5 mesos, compresos entre gener de 2012 i mitjans Juny de 2012. A mida que el projecte s'anava desenvolupant, es va denotar que l'estimació era massa optimista, a causa de no haver marges d'error per tenir la possibilitat de compensar la demora entre les fases realitzades.

A continuació es mostra una taula comparativa entre la temporització inicial i final.

Taques	Estimació prevista (dies)	Estimació real (dies)	Diferència (dies)
Estudi de Symfony	35	35	0
Configuració dels recursos	2	4	2
Anàlisis dels requeriments	7	14	7
Proves de la lògica de funcionament	2	4	2
Realització de plantilles web	4	4	0
Elaboració dels Bundles	5	5	0
Elaboració de la lògica de negoci de	30	65	35

l'aplicació			
Gestió d'entitats i permisos	10	17	7
Proves Unitàries	3	7	4
Accessibilitat de l'aplicació a Internet	2	3	1
Elaboració de la memòria	30	45	15
Total (dies) Total (hores)	130 650	202 1010	72 360

Taula 1: Dedicació total a l'elaboració de l'aplicació

Per a la realització del càlcul d'hores, s'ha estimat una dedicació diària de 5 hores. Com podem apreciar, el total d'hores que s'han dedicat a l'elaboració del projecte, ha estat força superior a la planificació inicial. Aquest augment es pot atribuir a diversos factors. El primer de tots és la fase d'aprenentatge del framework , la qual tot hi haver-hi realitzat un estudi i probes funcionals, no es podia arribar fins a un alt nivell de complexitat i tampoc preveure la gran dificultat que podria generar l'elaboració de l'aplicació. La informació desglossada que podem trobar a la xarxa, conforme exemples senzills i entenedors, on en certs aspectes son molt limitats, a conseqüència que la seva finalitat és didàctica, i sovint molts d'aquestos no tindrien cabuda en una aplicació web real.

Un altre aspecte el qual ha afavorit l'endarreriment de la planificació inicial, ha estat la fase d'elaboració de la lògica de negoci, on es desenvolupava tot el funcionament intern de l'aplicació. La manca d'exemples pràctics ha estat un impediment per poder assolir un ritme de treball continu, amb el motiu que sovint s'ha hagut de refer la implementació. Symfony2 ens presenta un nivell força complex d'abstracció, on el controlador de l'aplicació requereix i executa tota la funcionalitat de l'aplicació, i aquest fet a nivell personal ha estat un handicap a l'hora de desenvolupar-la.

2.2.2 Anàlisis econòmic

En aquesta secció es confecciona una simulació de costos del que suposaria la realització de l'aplicació en un àmbit fora del marc educatiu.

2.2.2.1 Cost de hardware

El cost dels recursos de hardware emprats en la realització de l'aplicació, es composen dels següents conceptes desglossats en la taula presentada a continuació:

Concepte	Quantitat	Import
Domini web .com	1	7,0682 € / any

Servei de hosting Personal Espai de disc 25 GB Tràfic il·limitat SQL Personal 1x50 MB	1	2,35 € / mes
Total		35,268 € / any

Taula 2: Cost de hardware

2.2.2.2 Cost de software

El cost dels recursos de software emprats en la realització de l'aplicació, són nuls deguts a la utilització d'eines de software lliure. El software emprat durant el procés d'elaboració de l'aplicació es presenta a continuació:

- Sistema operatiu Ubuntu Lucid 10.4.
- Framework Symfony2.
- Servidor web Apache.
- Gestor de dades MySQL.
- Gimp per l'edició d'imatges.

2.2.2.3 Cot de recursos humans

El cost dels recursos humans emprats en la realització de l'aplicació, es composen dels següents conceptes desglossats en la taula presentada a continuació:

Concepte	Quantitat	Import
Estudi del framework	195	0 €
Anàlisis dels requeriments i especificacions del client	75	300 €
Elaboració de la lògica de negoci, funcionalitats i disseny de l'estructura de l'aplicació	375	1500 €
Proves unitàries	35	140 €
Accessibilitat de l'aplicació a Internet	15	60 €
Total		2000 €

Taula 3: Cos de recursos humans

Per a la realització del càlcul de la quantitat, s'ha emprat una relació entre la dedicació d'hores i el concepte especificat. El import correspon a la tarificació total per concepte, el qual s'ha estimat a 4 € per hora, degut a que el ritme de treball no ha estat òptim, a

causa que s'ha hagut de dedicar gran part d'aquest a tasques d'investigació. Com a conseqüència s'ha assolit el salari d'un becari, el qual de mitjana s'aproxima a aquest import.

El motiu d'haver tarificat a com a nul el concepte d'estudi del framework, es conseqüència que és una tasca prèvia que el programador ha d'assolir amb anterioritat, i el seu cost no el pot assolir el client.

2.2.2.4 Cost total

Agrupant els costos presentats anteriorment, el cost global del projecte es desglossa en la present taula:

Concepte	Quantitat	Import
Recursos hardware	1	35,268 €
Recursos software	1	0 €
Recursos humans	1	2000 €
Total		2035,268 €

Taula 4: Cost total de l'aplicació

Com podem apreciar, el cost total ronda la mitjana d'una aplicació amb les especificacions mencionades.

3 Plataforma tecnològica

3.1 Symfony 2

Symfony és un framework desenvolupat per Fabien Potencier l'any 2003, el qual està basat en l'arquitectura MVC (model vista controlador), l'ORM (mapatge relacional d'objectes) de Doctrine i el framework Ruby on Rails.

3.1.1 Filosofia de Symfony 2

Symfony 2 és un complert framework dissenyat per optimitzar el desenvolupament d'aplicacions web. Separa la lògica de negoci, la lògica del servidor i la presentació de l'aplicació web. Proporciona utilitats i classes encaminades a reduir el temps de desenvolupament d'una aplicació web complexa. A més, automatitza les tasques més comuns, permetent al desenvolupador dedicar-se únicament a la lògica funcional.

Està desenvolupat completament en PHP 5, amb un motor de base de dades compatible amb la majoria de gestors. Es pot emprar tant en plataformes de software lliure (Unix) com en Windows.

Symfony va ser dissenyat per tal d'ajustar-se als següents requisits:

- Fàcil d'instal·lar i configurar en les principals plataformes.
- Fàcil d'emprar i manipular.
- Informació estructurada jeràrquicament.
- Basat en la premissa "convenir en comptes de configurar", on el programador tant sols ha de configurar allò que no es convencional (estàndard).
- Independent del sistema gestor de base de dades. La seva capa d'abstracció i l'ús de Doctrine, permeten canviar fàcilment el sistema gestor de base de dades en qualsevol fase del projecte.
- Utilització de la programació orientada a objectes, d'aquí el motiu pel qual es indispensable PHP 5.
- Arquitectura basada en el MVC, tot i emprar la seva pròpia variant , com es el cas de l'abstracció de la base de dades, el controlador frontal i les accions.

- Preparat per a la realització d'aplicacions empresarials i adaptable a les politiques i arquitectures pròpies de cada empresa, a més de la seva gran flexibilitat que permet canviar el seu model en la fase de desenvolupament.
- Codi entenedor i estructurat, el qual permet la realització estàndard de documentació per part del desenvolupador.
- Fàcil d'estendre, cosa que permet la integració mitjançant les biblioteques d'altres fabricants.
- Una potent línia d'accions que faciliten la generació de codi estàndard, factor que contribueix a estalviar temps i millorar l'estructuració.
- Documentació clara i esquematitzada, la qual facilita la introducció al seu software.
- Permet la internacionalització de la interfície, les dades i els continguts de localització.
- La presentació empra templates i layouts, els quals poden ser construïts per dissenyadors d'HTML que no posseeixin coneixents previs del framework.
- Els formularis suporten la validació automàtica, el qual assegura un correcte emmagatzemament de les dades i una millor experiència per l'usuari.
- La utilització de cache redueix l'ús d'ample de banda i la càrrega al servidor.
- Facilitat per suportar autenticació i credencials, per tal de brindar la creació d'àrees restringides .
- L'encaminament i les URL intel·ligents fan amigables les direccions de les pàgines de l'aplicació.
- La generació de bundles, facilita l'estandardització d'accions úniques a les diferents àrees de negoci.

Symfony 2, és la versió més recent del framework, el qual suposa un canvi radical tant en l'arquitectura com en la filosofia de treball respecte a les versions anteriors. La seva arquitectura interna està completament desacoblada (dividida en bundles), cosa que permet reemplaçar o eliminar fàcilment aquelles parts que no encaixin en la nostra aplicació, millorant així el rendiment d'aquesta, com a resultat es redueix la càrrega computacional amb la que aquest ha de treballar.

En comparació amb la resta de frameworks, es el que més funcionalitats incorpora i el que disposa de major flexibilitat i llegibilitat.

3.1.2 Arquitectura MVC

Symfony 2, centra el seu funcionament intern en la popular arquitectura Model – Vista – Controlador (MVC). No obstant, aquest sol proporciona eines per modelitzar la part del controlador i de la vista. La part del model, es feina del programador dissenyar-la, tot i que existeixen infinitat de llibreries per integrar els ORMS més coneguts. En el nostre cas en particular, hem emprat el ORM Doctrine 2, amb suport per a twigs. Observem la figura 9:

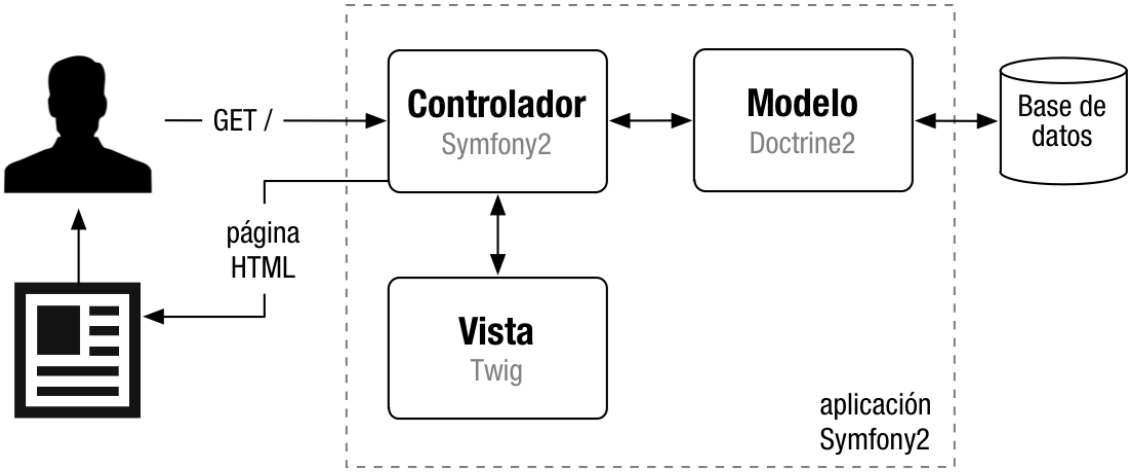


Figura 9: Funcionament de l'arquitectura MVC

Quan l'usuari sol·licita qualsevol URL de l'aplicació, internament succeeix el següent:

- El sistema d'encaminament determina quin **controlador** està associat a la pàgina seleccionada.
- Symfony 2 executa el controlador associat a aquesta. Un controlador l'hem d'entendre com una classe PHP, on podem definir totes les funcionalitats del codi especificat.
- El controlador sol·licita al **model** les dades especificades. El model l'hem d'entendre com una classe PHP especialitzada en obtindre informació, normalment d'una base de dades.
- Amb les dades retornades pel model, el controlador sol·licita a la **vista** la creació d'una pàgina mitjançant una plantilla que interactuï amb les dades del model.
- El controlador entrega al servidor la pàgina creada per la vista.

Tot i que podem arribar a realitzar aplicacions molt complexes, el funcionament intern sempre segueix el mateix patró:

1. El **controlador** demana i direcciona.

2. El **model** busca la informació sol·licitada.
3. La **vista** crea pàgines mitjançant les plantilles i les dades.

3.1.2.1 Controlador

El controlador és la part de l'aplicació destinada a contenir tota la *lògica de negoci*, això significa que aquest s'encarrega de gestionar tota la funcionalitat de l'aplicació.

3.1.2.2 Model

El model és la part de l'aplicació la qual s'encarrega de registrar tota la informació recuperada de la base de dades en les diferents entitats. Com s'ha esmentat anteriorment, com a modelatge de la informació s'ha emprat l'ORM Doctrine 2. Les funcions bàsiques que aquesta ha de satisfer (acció d'emmagatzematge, consulta i eliminació), es realitzen mitjançant l'objecte especial *entity manager.*

3.1.2.2.1 Entity Manager

Symfony 2 crea aquest objecte automàticament i el posa a la nostra disposició a través del *contenidor d'injecció de dependencies*. La forma de definir-lo es la següent:

- `$em = $this -> getDoctrine() -> getEntityManager();`
- `$producto = $em -> getRepository('ProductoBundle: Producto') -> find ( 1);`

La primera funció vincula l'entity manager amb la base de dades. La segona funció, obté totes les dades referents a l'entitat article. El mètode find(), busca entitats mitjançant la seva clau primària. Com es habitual, el nom de la entitat s'indica mitjançant la *notació bundle*, sent la primera part el nom del bundle i la segona el nom de la classe de l'entitat.

Un dels avantatges d'emprar l'*entity manager*, és la dràstica reducció de les consultes a la base de dades, cosa que comporta una millor optimització de la nostra aplicació. Però on radica la seva utilitat, es en el següent context:

- Imaginem que estem accedint a la taula vendes, on aquesta taula emmagatzema l'ID de l'usuari que ha realitzat la compra, i volem mostrar el nom i cognoms de l'usuari.
- Per tal de modelitzar la petició anterior, fem servir el següent codi:

- Controlador:
  - $em = $this -> getDoctrine() -> getEntityManager();
  - $venta = $em -> getRepository ('ProductoBundle:Venta') -> findAll();
- Plantilla:
  - {% for venta in venta %}

    {{ venta.usuario.nombre }}

    {{ venta.usuario.apellidos }}

    {% endfor %}

Ens podríem preguntar si el codi presentat anteriorment funciona. Tot i que resulta força abstracte d'entendre, la resposta és afirmativa. La clau per entendre-ho, resideix en la característica *lazy loading* de Doctrine 2. Aquest comportament fa que Doctrine 2 busqui automàticament en la base de dades qualsevol informació que es sol·liciti i no estigui disponible. La instrucció {{ venta.usuario.apellidos }} comporta que Doctrine 2 realitzi una consulta a la base de dades per tal d'obtenir tots els paràmetres de l'usuari l'ID del qual es referència en la taula de vendes. Tot i que el *lazy loading* és completament transparent pel programador i sembla una utilitat brillant, pot penalitzar sèriament el rendiment de l'aplicació. El bucle for del codi anterior, podria realitzar centenars de consultes a la base de dades, una per cada usuari les dades del qual no s'hagin trobat amb anterioritat. Per aquest motiu es recomanable no abusar-hi.

3.1.2.2.2 Contenidor d'injecció de dependencies

Crea tots els objecte de la nostra aplicació. Aquest està dotat de la relació entre les classes i la configuració necessària per instanciar-les correctament. Observem el següent exemple:

```
$contenedor = new Contenedor();
$logger = $contenedor->getLogger();

$logger->info('...');
```

Figura 10: Contenidor de dependencies

L'aplicació simplement demana un *logger* al contenidor, sense la necessitat d'amoïnar-nos per com el crea, o com es realitza la instància o la configuració de les classes i el posa a la nostra disposició d'igual forma que una *funció anònima*.

3.1.2.2.3 Funcions anònimes

Les funcions anònimes, també anomenades *closures*, són funcions sense nom que normalment s'empren per crear un *callback* (crida a les funcions del sistema). El codi font de Symfony 2 realitza un us extensiu d'aquestes funcions. Observem la figura 11 (extracció de la classe Symfony/Component/Console/Aplication.php):

```
public function renderException($e, $output)
{
    $strlen = function ($string) {
        if (!function_exists('mb_strlen')) {
            return strlen($string);
        }

        if (false === $encoding = mb_detect_encoding($string)) {
            return strlen($string);
        }

        return mb_strlen($string, $encoding);
    };

    // ...

    $len = $strlen($title);
```

Figura 11: Funcions anònimes

La variable $strlen emmagatzema una funció anónima que calcula la longitud d'una cadena de text. Aquesta funció s'adapta a les característiques del sistema en el que s'executa, emprant mb_strlen() o strlen() per determinar la longitud de la cadena.

El codi intern d'una funció anònima no té accés a cap part de l'aplicació. Totes les variables que precisin codi, s'han d'importar mitjanant la instrucció reservada use.

3.1.2.3 Vista

La vista és la part de l'aplicació encarregada de generar la interfície gràfica. Aquesta es generada a partir de les diferents plantilles les quals la composen. Per a la realització de les plantilles, s'ha emprat els llenguatge de programació web següents:

- HTML, per definir l'estructura i el contingut de l'aplicació.
- CSS, com a fulla d'estils, per tal de dotar de personalitat pròpia a l'aplicació.
- JavaScript, per tal de dotar l'aplicació d'accions compatibles amb Google Maps.
- Twig, com a motor de contingut dinàmic de les plantilles.

3.1.2.3.1 Twig

Es un motor de llenguatge de plantilles per a PHP molt ràpid i eficient. La seva sintaxi ha estat dissenyada amb la premissa de creació de plantilles més concises i major facilitat per llegir-les i escriure-les. Observem el següent exemple:

```
{% if usuario is defined %}
  Hola {{ usuario.nombre }}
  hoy es {{ 'now' | date('d/m/Y') }}
{% endif %}
```

Figura 12: Extracte del llenguatge twig

```
<?php if (isset($usuario)): ?>
    Hola <?php echo htmlspecialchars($usuario->getNombre(), ENT_QUOTES,
'UTF-8') ?>
    hoy es <?php $hoy = new \DateTime(); echo $hoy->format('d/m/Y');  ?>
<?php endif; ?>
```

Figura 13: Extracte del llenguatge PHP

El codi presentat en les dues figures anteriors realitza la mateixa funcionalitat, però com podem apreciar, el llenguatge twig es molt més esclaridor i entenedor. Per tal de delimitar el seu codi, aquest empra tres etiquetes:

- {{ i }}, per tal de mostrar el valor d'una variable.
- {% i %}, per tal d'afegir la lògica en la plantilla.
- {# i #}, per tal d'incloure un comentari.

3.1.3 Namespaces

Un namespace és un contenidor abstracte el qual agrupa de forma lògica varis símbols e identificadors als quals ha de tenir accés el controlador. L'hem d'entendre com una direcció lògica a totes les classes PHP. En la pràctica, aquestos s'empren per estructurar el codi font de l'aplicació. Totes les classes PHP desenvolupades en l'aplicació emprem namespaces, motiu pel qual és essencial realitzar una menció prèvia a aquestos.

Abans de l'existència dels namespaces, el nom de les classes havia de ser únic per a cada acció, cosa que sovint denotava una mala estructuració i confusió per part del programador per arribar a determinar l'acció a la qual s'havia de respondre. Gracies als namespaces, dos o més classes de l'aplicació poden compartir el seu nom. L'únic requisit és que el nom dels namespaces siguin diferents, de forma que l'aplicació sàpiga en tot moment quin és el que s'està emprant.

Posem per exemple que volem emprar la classe PHP anomenada UsuarioType.php ubicada al directori Symfony_Paradeta/src/Paradeta/UsuarioBundle/Form/Frontend/ Per tal d'accedir-hi i emprar-la en la classe DefaultController.php, hem de referenciar el seu namespace e importar-la mitjançant la instrucció *use*, de la següent forma:

```php
<?php

namespace Paradeta\UsuarioBundle\Controller;

use Paradeta\UsuarioBundle\Form\Frontend\UsuarioType;
```

Figura 14: Referència a la classe UsuarioType.php

La instrucció *use* referència el namespace de la classe o directoris de classes que es troben disponibles per emprar al controlador d'altres classes. Quan utilitzem una classe especifica, és més òptim importar la classe en comptes del seu directori mitjançant la instrucció *use* i el nom de la classe sense el sufix .php

3.1.4 Anotacions

Son "accions" introduïdes en el propi codi de l'entitat mitjançant el mapatge ORM de Doctrine 2. El controlador les interpreta com simples comentaris, però gracies a la llibreria de Doctrine2, aquestos influeixen en l'execució de codi. Observem el següent exemple:

```php
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;

// ...

/**
 * @ORM\Column(type="string")
 * @Assert\NotBlank()
 */
protected $nombre;
```

Figura 15: Validació del nom mitjançant la notació ORM

Just a sobre de la declaració $nombre s'afegeix un comentari de varies línees. El contingut d'aquest no es trivial, perquè proporciona informació clau sobre la propietat. La instrucció @ORM\Colum(type="string") defineix el tipus de columna de base de dades en la qual s'emmagatzema la propietat, i la instrucció @Assert\NotBlank() assegura que aquesta propietat no es deixi buida.

3.1.5 Llenguatge YAML

Els arxius de configuració de Symfony 2 es poden implementar mitjançant el llenguatge PHP, XML ó YAML. La diferencia entre la utilització d'aquestos no prima en el rendiment (a causa que tots ells es transformen en codi PHP abans de l'execució de l'aplicació), sinó en el temps que el programador ha de dedicar per tal de confeccionar instruccions:

- PHP: Es el llenguatge més laboriós d'escriure la configuració, però és el que major flexibilitat proporciona.
- XML: Es el llenguatge més caòtic i amb major llargada final, però el seu gran avantatge es la validació en temps de producció.
- YAML: Probablement és el més equilibrat, a causa que és molt més concís que XML i força flexible. El seu gran inconvenient és que no es pot validar automàticament, cosa que implica que la gran majoria d'errors els advertim en temps d'execució.

El llenguatge YAML, es composa dels següents conceptes bàsics:

- Mai hem d'emprar el tabulador, per tal d'estructurar el codi hem de fer servir quatre espais en blanc.
- La informació s'indica mitjançant el par *clau: valor*. Si la clau o el valor té espais en blanc, es tanquen mitjançant comes simples o dobles.
- Els *arrays normals* s'indiquen mitjançant els claudàtors [i], en canvi, els *arrays associatius* s'indiquen mitjançant les claus { i }.
- Els comentaris es prefixen mitjançant el caràcter #.

Observem la configuració de seguretat per defecte de Symfony 2 (extracció de la classe app/Config/Security.yml):

```
security:
    encoders:
        Symfony\Component\Security\Core\User\User: plaintext

    role_hierarchy:
        ROLE_ADMIN:       ROLE_USER
        ROLE_SUPER_ADMIN: [ROLE_USER, ROLE_ADMIN, ROLE_ALLOWED_TO_SWITCH]

    providers:
        in_memory:
            users:
```

```
user:  { password: userpass, roles: [ 'ROLE_USER' ] }
admin: { password: adminpass, roles: [ 'ROLE_ADMIN' ] }
```

Figura 16: Configuració de seguretat per defecte

La paraula security de la primera línea és la clau principal d'aquest arxiu YAML, a partir del qual es defineixen la resta de claus i valors. Observem com les claus de segon nivell (encoders, role_hierarchy i providers) van precedides de quatre espais en blanc.

Dins de la clau role_hierarchy, l'array de valors de la clau ROLE_SUPER_ADMIN es defineix emprant la notació tradicional de claudàtors. En canvi, si ens fixem en la clau users, podem observar com definim un array associatiu de valors.

Tot aquest conjunt estableix una jerarquia de classe en un arxiu YAML. Al processar l'arxiu YAML, Symfony 2 transforma el codi anterior en el següent array PHP:

```
array(
    'security' => array(
        'providers' => array(
            'in_memory' => array(
                'users' => array(
                    'user'  => array(
                        'password' => 'userpass',
                        'roles'    => array('ROLE_USER')
                    ),

                    'admin' => array(
                        'password' => 'adminpass',
                        'roles'    => array('ROLE_ADMIN')
                    )
                )
            )
        )
    )
)
```

Figura 17: Composició de l'array de seguretat PHP

3.1.6 Entitats

El codi PHP de les aplicacions de Symfony 2 no interactua directament amb la base de dades. Per aquesta motiu la informació no es gestiona mitjançant sentencies SQL, sinó amb objectes PHP. Aquestos objectes es denominen tècnicament *entitats*. Definir les entitats de l'aplicació radica en traduir la informació de les taules de la base de dades a classes PHP.

3.1.7 Llenguatge DQL

Doctrine 2 defineix un llenguatge propi anomenat *DQL* mitjançant el qual es poden realitzar consultes a la base de dades. La seva sintaxis es molt similar al llenguatge SQL, però el seu funcionament intern és radicalment diferent. DQL realitza consultes sobre objectes PHP, mentre SQL realitza consultes sobre taules.

El llenguatge DQL permet realitzar consultes de tipus Select, Update i Delete. Les consultes de tipus Insert no estan permeses, a causa que tota la nova informació s'ha de crear mitjançant el mètode persist() de l'*entity manager*. Observem la següent consulta:

- $em = $this -> getDoctrine() -> getEntityManager();
- $consulta = $em -> createQuery('SELECT p FROM ProductoBundle: Producto p WHERE p.preu < 20 ORDER BY p.nom ASC');
- $producto = $consulta -> getResult();

La consulta presentada, es divideix en quatre seccions:

- SELECT p: La secció SELECT indica quina informació retorna la consulta. En aquest cas, es retorna tota la informació de l'objecte Producto, motiu pel qual simplement s'indica l'identificador de l'entitat emprada en el FROM. En el cas que sol ens interessés el nom de l'article i el seu preu, s'indicaria mitjançant:
  - 'SELECT p.nom, p.preu FROM ...'
- FROM ProductoBundle: Producto p: La secció FROM indica les entitats sobre les quals es realitza la consulta. La entitat s'especifica mitjançant la *notació bunde* (nom-bundle: nom-entitat). Després de l'entitat, s'inclou un identificador (p en el nostre cas en particular), el qual s'utilitza per identificar-la (normalment s'elegeix l'inicial del nom de l'entitat).
- WHERE p.preu < 20: La secció opcional WHERE especifica les condicions que han d'acomplir els objectes per tal de ser inclosos en el resultat de la consulta.
- ORDER BY p.nom ASC: L'última secció també és opcional, i afegeix noves condicions o modifica l'ordre en que es retornen els resultats.

A la secció 4.2.6 , s'amplien els continguts sobre la modelització de la informació de la base de dades introduïda en aquest apartat.

3.1.8 Bundles

Un bundle és un directori el qual conté tot tipus d'arxius dins d'una estructura jerarquitzada. Aquestos, solen englobar totes les classes PHP i arxius web. No obstant, no existeix cap tipus de restricció pertinent al que es pot emmagatzemar a l'interior d'un bundle.

No hi ha definit un estàndard sobre la capacitat i els arxius que aquest ha de contindre. Alguns programadors recomanen generar un únic bundle gegantí per tal d'arribar a reutilitzar-lo en futures revisions. D'altres, recomanen separar-los seguint la lògica de l'aplicació. Per desenvolupar la present aplicació, s'ha decidit crear tres bundles seguint la lògica de permisos de l'aplicació:

- Frontend:
  - ProductoBundle: Contindrà tots els arxius necessaris per generar els continguts pertanyents a la part pública (portada, carta de sidres i vins, etc.).
  - UsuariosBundles: Contindrà tots els arxius necessaris per generar els continguts pertanyents a la part privada (compres realitzades, perfil, etc.).
- Backend:
  - BackendBundle: Contindrà tots els arxius necessaris per generar els continguts pertanyents a la part d'accés restringit (modificació del catàleg de productes, visualització de compres, etc.).

La distribució presentada, diferencia la part pública de la privada, de la mateixa forma que succeïa amb versions anteriors del framework.

3.1.9 Jerarquia de directoris

La composició de Symfony 2 es troba organitzada per una jerarquia de directoris; la següent taula ens il·lustra una visió global d'aquesta:

Directori	Propòsit
App	Conté tots els arxius de configuració, la *caché*, els *logs* i els recursos globals.
app/config/	Emmagatzema tots els arxius de configuració de l'aplicació.
app/cache/	Conté tots les arxius generats per les classes, plantilles, entitats, etc. Junt amb el directori app/logs es l'únic sobre

	el que Symfony 2 ha de posseir permisos d'escriptura.
app/logs	Conté tots els arxius generats en l'entorn d'execució.
src/	Emmagatzema tot el codi font del propi projecte. En aquest directori es on seran accessibles els bundles de l'aplicació.
src/Paradeta/ProductoBundle/Resources	Contindrà les plantilles de cada bundle de l'aplicació.
src/Paradeta/ProductoBundle/Controller	Contindrà el controlador de cada bundle de l'aplicació.
src/Paradeta/ProductoBundle/Entity	Contindrà les entitats de cada bundle de l'aplicació.
vendor/	Conté tot el codi font de Symfony 2 i totes les llibreries externes.
web/	És l'únic directori públic del projecte. Contindrà tots els arxius web (CSS, JavaScript e imatges) i els controladors frontals de l'aplicació (app.php i app_dev.php).

Taula 5: Jerarquia de directoris

3.1.10 Encaminament

El sistema d'encaminament transforma les URL en controladors. Concretament, determina quin és el controlador que s'ha d'executar per a cada URL sol·licitada per l'usuari. L'arxiu principal d'encaminament de Symfony 2 es app/Config/Routing.yml, on la nomenclatura a seguir és la següent:

```
_portada:
    pattern: /
    defaults: { _controller: ProductoBundle:Default:portada}
```

Figura 18: Configuració d'encaminament

Tal i com s'ha introduït anteriorment, cadascuna de les rutes de l'aplicació es defineixen mitjançant un nombre únic (*portada* en aquest cas en particular), una expressió regular (pattern) que han d'acomplir les URL (en el cas de la portada es simplement /) i el nom del controlador el qual s'executa al respondre la petició (mitjançant la clau especial anomenada _controller).

Symfony2 interpretar la petició del controlador com una referència, mitjançant la *notació bundle*, a l'acció portadaAction() de la classe DefaultController.php, la qual es ubicada a src/Paradeta/ProductoBundle/Controller/.

Els enllaços de les plantilles twig s'inclouen mitjançant la funció path() a la qual se li referència com primer argument el nom de la ruta. D'aquesta forma els enllaços de les diferents plantilles es generen dinàmicament en el mateix instant en que aquesta es

representa. El principal avantatge d'aquesta técnica, radica en que si es modifica algun dels arxius d'encaminament, tots els arxius de l'aplicació s'actualitzen instantàniament sense la necessitat de modificar tot el seguit de plantilles.

3.1.11 Entorns d'execució

Una aplicació de Symfony 2 no sol es composa del codi font inclòs en el directori /src, sinó que també es defineix mitjançant els arxius de configuració amb els quals es controla el seu comportament.

Quan la URL d'una pàgina inclou el terme app_dev.php, l'aplicació s'executa en l'entorn dev o *entorn de desenvolupament*. Les funcionalitats d'aquest entorn estan pensades per a programadors. Executar d'aquesta forma l'aplicació es molt més lent, però totes les pagines inclouen la *barra de depuració web*.

Figura 19: Barra de depuració web

Aquesta ens mostra tot un seguit d'informació referent al controlador i a les accions que aquest realitza. El més interessant és el log d'errors, el qual ens indica el nombre d'errors produïts en temps d'execució. També es mostra el nombre total de consultes realitzades a la base de dades (informació molt útil per poder determinar si es realitza un mal ús de la configuració *lazy loading*).

De la mateixa forma, quan una URL d'una pàgina inclou el terme app.php, l'aplicació s'executa en l'entorn prod o *entorn de producció*. Aquest és l'entorn en que l'aplicació s'ha d'executar en el servidor de producció. Les pagines generades no inclouen cap tipus d'informació útil per al programador, però en canvi, l'aplicació s'executa amb la major brevetat possible.

3.1.12 L'objecte Request

Symfony2 encapsula tota la informació de la petició de l'usuari en un objecte de tipus *Request*. D'aquesta manera, es centralitza tota la informació de les variables globals PHP.

En un controlador, l'objecte de la petició s'obté mitjançant el mètode getRequest(). La classe Request inclou un mètode anomenat get(), el qual conté el valor del paràmetre el nom del qual s'instància per referència. Aquest paràmetre segueix sempre la matei-

xa lògica de funcionament, $_GET, $_SERVER['PATH_INFO'], i $_POST:

```
class DefaultController extends Controller
{
    public function portadaAction()
    {
        $peticion = $this->getRequest();
    }
}
```

Figura 20: Petició getRequest()

El mètode getRequest() presentat a la figura superior, només funciona per als controladors per defecte de Symfony 2 que hereten la classe Controller(). Aquest controlador base empra totes les funcions de sistema. Si el nostre controlador no hereta de Controller(), podem obtindre l'objecte a través del servei request del contenidor d'injecció de dependencies:

- $peticio = $this -> container -> get('request')

La classe Request també inclou diversos mètodes relacionats amb la URL e informació de la petició. Imaginem que l'usuari ha sol·licitat l'accés a la pàgina http://localhost/app_dev.php/sidra:

Mètode	Descripció
getMethod()	Retorna el nom en majúscules del mètode emprat en la petició (GET, POST, DELETE, ...).
getRequestUri()	Retorna la URI de la petició: /app_dev.php/sidra.
getUri()	Retorna la URI *normalitzada* de la petició: http://localhost/app_dev.php/sidra.
get Scheme()	Retorna l'esquema emprat: http o https.
getHost()	Retorna el *host* de l'aplicació: localhost.
getPort()	Retorna el port al que es realitza la petició: 80 o 443.
getQueryString()	Retorna la *query string* normalitzada de la URI: sidra.
getScriptName	Retorna el nom del *script* PHP que s'executa: app_dev.php.
getClientIP()	Retorna la direcció IP de l'usuari que ha realitzat la petició. Si l'usuari empra un *proxy*, s'inicialitza a *true* el valor del mètode.
getLanguages()	Retorna un array amb el codi dels idiomes preferits per l'usuari: array('en_US', 'en', 'es').

Taula 6: Mètodes de la classe get()

3.1.13 L'objecte Response

Symfony2 encapsula en un objecte del tipus Response() tota la informació necessària per generar la pàgina que s'entrega a l'usuari com a resultat de la seva petició. La ma-

joria dels controladors finalitzen la seva execució invocant el mètode render() per generar una pàgina fent us de la plantilla instanciada.

En realitat, el mètode render() és una drecera definida en la classe Controller() de la qual hereten per defecte els controladors de Symfony 2. Emprant el sistema de plantilles definit en l'aplicació, aquest crea un objecte de tipus Response() que Symfony 2 converteix en una pàgina HTML, la qual és retornada a l'usuari.

3.1.14 Funcionament intern

Un cop especificades totes les accions a les quals ha de donar servei Symfony 2, detallarem el procés complert des de que l'usuari sol·licita una URL fins que el servidor web li entrega una pàgina com resposta.

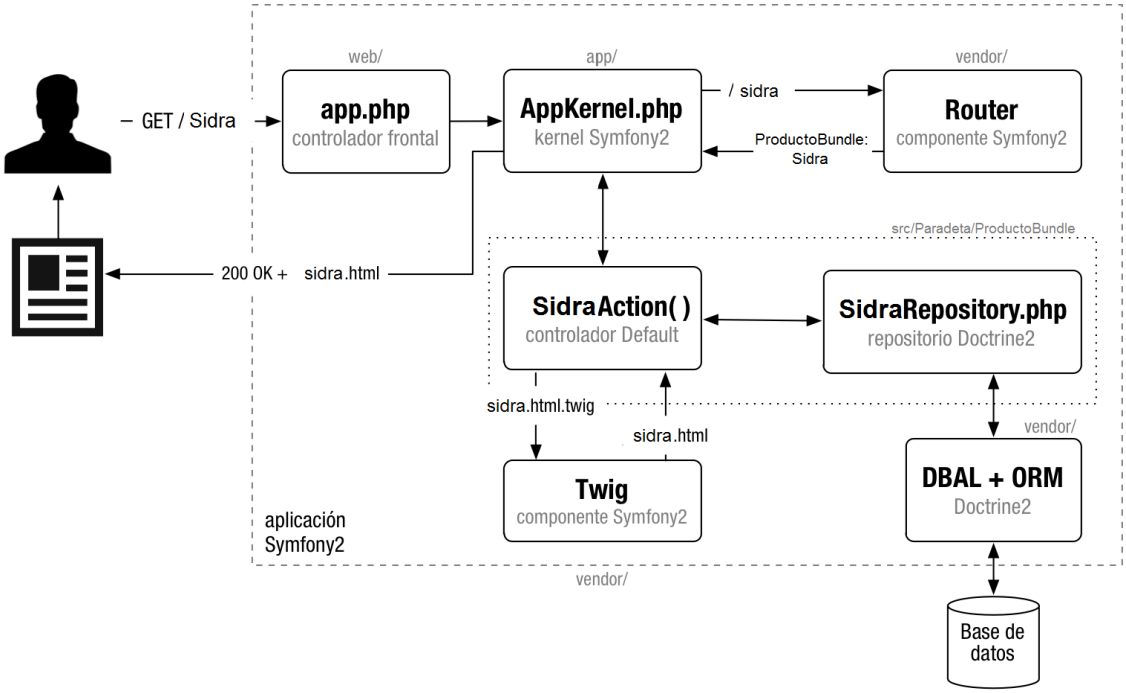


Figura 21: Funcionament intern de Symfony 2

L'usuari sol·licita la carta de sidres a través de la URL pàgina http://localhost/app.php/sidra. Les regles definides a l'arxiu .htaccess del directori /web de l'aplicació, redirigeixen la petició al *script* app.php. Aquest arxiu s'anomena **controlador frontal** i és l'únic punt d'entrada a l'aplicació.

El controlador frontal crea el *Kernel* de l'aplicació mitjançant la instància de la classe app/AppKernel.php a la qual se li referència el valor *prod* com nom de **l'entorn d'execució**. El kernel registra tots els *bundles* actius de l'aplicació i els emmagatzema a l'arxiu de configuració app/config/config_prod.yml. Al mateix temps, aquest importa l'arxiu app/config/config.yml el qual carrega les rutes de app/config /routing.yml.

Seguidament, el controlador frontal crea un objecte de tipus Request on afegeix tota la informació de la petició i la referència al *kernel*. El processament de la petició comença notificant les accions de sistema i continua buscant al component d'**encaminament** quin controlador s'ha d'executar per atendre a la petició de l'usuari.

Un cop localitzat el **controlador** i l'acció corresponent a la URL solicitada, s'executa el seu codi. Si és necessari, el controlador fa ús del **model** per tal de cercar la informació en la base de dades. Normalment, aquesta cerca es realitza mitjançant un **repositori** propi creat per a les entitats de Doctrine 2.

Finalment, si el controlador retorna un objecte de tipus Response, el *kernel* el transforma en la pàgina que es retorna a l'usuari. Si el controlador retorna el nom d'una plantilla, el *kernel* fa ús de la **vista** per renderitzar la plantilla segons el motor configurat en l'aplicació (Twig o PHP). L'execució finalitza retornant a l'usuari la pàgina creada a partir de la plantilla.

3.2 Google Maps

Google Maps és una potent eina de geolocalització de coordenades cartesianes, creat per l'empresa *Where 2 Technologies* amb una posterior adquisició per part de Google l'any 2004. Ofereix un servei d'imatges sobre mapes desplaçables, així com fotografies per satèl·lit, la ruta entre dos ubicacions e imatges d'*street view*. Tot el seu codi es composa del llenguatge *Javascript* i *Ajax*, orientats a objectes.

Originàriament, va ser un projecte *"tancat"* als desenvolupadors de software externs a l'empresa. Atent a la gran demanda i el gran volum de negoci que podria arribar a generar, va impulsar l'empresa Google a generar una API pública, oferint la possibilitat d'integrar tots els seus serveis en portals externs a Google Maps, mitjançant la interacció amb el seu servidor. La versió més actual d'aquesta es la 3, la qual s'ha fet servir en la present aplicació.

3.2.1 Interacció aplicació – Google Maps

L'elmet fonamental el qual vincula l'aplicació amb el propi mapa es l'API. Aquesta la podem definir com el nexe d'unió entre el desenvolupador i el servidor de Google.

La manera d'incloure-la és mitjançant un *script* de tipus *text/javascript*.

```
<script type="text/javascript"
    src="https://maps.googleapis.com/maps/api/js?sensor=false&language=es"">
</script>
```

```
<script type="text/javascript">
```

Figura 22: Definició de l'API de Google Maps

- La URL http://maps.google.com/maps/api/js permet accedir a la ubicació d'un arxiu *JavaScript*, el qual és l'encarregat de generar tots els símbols i definicions necessàries per tal d'emprar de forma adequada l'API.
- El paràmetre sensor, indica que l'aplicació no empra un sensor, com per exemple un GPS, per determinar la ubicació de l'usuari.
- El paràmetre language, indica el idioma de l'àrea de treball del servei Google Maps.

3.2.2 Opcions de mapa

Abans d'inicialitzar un mapa, s'ha de crear un objecte Map options, el qual ha de contenir les variables d'inicialització corresponents. Aquest objecte no es construeix , sinó que es crea com un literal.

```
function initialize() {

<!--Variables de configuraci xF3n-->

var myOptions = {
        zoom: 18,
        center: new google.maps.LatLng(42.407286, 0.741794),
        mapTypeId: google.maps.MapTypeId.ROADMAP
    };
```

Figura 23: Variables d'inicialització del mapa

- El paràmetre zoom, indica els nivells de zoom els quals establiran el mosaic d'imatges a mostrar.
- El paràmetre center, indica la ubicació predefinida, la qual es mostrarà a l'usuari. Les coordenades s'han d'indicar mitjançant el mètode google.maps .LatLang(latitud, longi-tud).
- El paràmetre mapTypeId, indica quin és el tipus de mapa a mostrar. Aquest es referència mitjançant el mètode google.maps.MapTypeId.ROADMAP. La variable ROADMAP exposa que els mosaics es mostren en 2D. També pot acceptar altres valors tals com SATELLITE, la qual ens mostra una imatge per satèl·lit.

3.2.3 L'objecte elemental

La classe *JavaScript* que representa el mapa és Map. Cada objecte d'aquesta classe, defineix un únic mapa, tot i que es poden instanciar tants objectes com es vulguin.

```
var map = new google.maps.Map(document.getElementById('map_canvas'),
    myOptions);
```

Figura 24: Creació de l'objecte mapa

- El codi anterior permet definir la variable map com un objecte de la classe Map().
- Creem una nova instància d'aquesta classe mitjançant l'operador new de *JavaScript*.
- El paràmetre map_canvas, especifica l'element HTML el qual serà emprat per identificar el contenidor. Aquest es referència mitjançant el mètode document.getElementBy-Id().

3.2.4 Esdeveniments

El codi JavaScript presentat es troba *orientat a esdeveniments*, cosa que comporta que aquest ha de respondre a les interaccions amb l'usuari. Trobem dos tipus d'esdeveniments:

- Els esdeveniments d'usuari (com per exemple el *click* del ratolí) es propaguen des de el DOM fins l'API de Google Mpas.
- Les notificacions de canvi d'estat de MVC reflecteixen els canvis en objectes de l'API de Google Maps.

```
<!--Referenciamos las variables a mostrar en el mapa-->

    google.maps.event.addListener(marker, 'click', function() {
        infowindow.open(map,marker);
    });



    google.maps.event.addListener(directionsDisplay, 'directions_changed', function() {
        computeTotalDistance(directionsDisplay.directions);
    });

}
```

Figura 25: Esdeveniments d'usuari

Cada objecte de l'API de Google Maps exporta una determinada quantitat d'esdeveniments. Els programes interessats en determinats esdeveniments, registren detectors

d'esdeveniments de JavaScript i executen el codi al rebre'ls mitjançant el registre de controladors AddListener() en l'espai de nom google.maps.event.

3.2.4.1 Esdeveniments en l'interfície d'usuari

Alguns dels objectesde l'API de Google Maps es troben dissenyats amb la finalitat de respondre a les accions d'usuari, donats com el ratolí o teclat. Per exemple, l'objecte google.maps.Marker pot detectar els següents esdeveniments d'usuari:

- Click
- Dblclick
- Mouseup

Aquestos poden presentar l'aspecte d'esdeveniments DOM estàndards, però en realitat, formen part de l'API de Google Maps. Treballant sobre la premissa que els diferents navegadors disponibles implementen diversos models d'esdeveniments DOM, l'API de Google Maps ofereix la possibilitat de detectar-los i respondre a aquestos sense la necessitat de gestionar les diverses peculiaritats de cada plataforma. Els esdeveniments solen incloure arguments els quals destaquen l'estat de la interfície d'usuari (com per exemple, la posició del ratolí).

3.2.4.2 Canvis d'estat del MVC

Els objectes MVC acostumen a contenir l'estat. Quan canvia la propietat d'un objecte, l'API activa un esdeveniment indicant que aquesta ha canviat. Per exemple, l'API activa un esdeveniment directions_changed en un en el moment que l'usuari indica que s'ha de traçar una ruta entre dos punts. Per interceptar aquestos canvis d'estat, es registra al controlador d'esdeveniments AddListener() el mètode d'espai de nom event.

3.2.5 Superposicions

Les superposicions són objectes del mapa les quals estan vinculades a coordenades de latitud i longitud, com a conseqüència són mòbils i sensibles al nivell de zoom. Aquestos, són objectes afegits els quals interactuen amb el mosaic que forma el mapa. Google Maps incorpora diverses superposicions, entre les quals destaca:

- Marker: Són marcadors referenciats mitjançant coordenades. Aquestos s'instancien mitjançat l'objecte Marker().

- Finestra d'informació: Aquesta és una instància de la classe InfoWindow(), la qual mostra, mitjançant un globus emergent, una cadena de text.
- Polilínees: Aquesta representa una sèrie ordenada d'ubicacions. És instanciada mitjançant la classe directionsDisplay().

3.2.5.1 Marcadors

Els marcadors identifiquen ubicacions en el mapa. De forma predeterminada, empren una icona estàndard, tot i que es pot establir un personalitzat mitjançant l'execució de l'objecte icon.

```
<!--Logo a mostrar en el marker-->

var logo_restaurante = new google.maps.MarkerImage('paradeta.png');


<!--Modficamos la posici[xF3]n inicial poruqe el tama[xF1]o del logo es consid

var posicion_restaurante = new google.maps.LatLng(42.407151, 0.741867);


<!--Texto a mostrar en el marker-->

    var contentString =
    '<div id="content">'+
            '<h1 id="firstHeading" class="firstHeading">La Paradeta</h1>'+
            '<div id="bodyContent">'+
                '<p> ' +
                'Calle ciutat de Lleida, n&uacute;mero 5 <br/>'+
            'El Pont de Suert, 25520 <br/>'+
            'Tel&eacute;fono: 666666666 <br/>'+
            'Latitud: 42.407286 - Longitud: 0.741794'+
                '</div>'+
        '</div>';



<!--Variable asociada a la carga de la informaci[xF3]n-->

    var marker = new google.maps.Marker({
        position: posicion_restaurante,
        icon: logo_restaurante,
        map: map,
        title: 'La Paradeta'
    });
```

Figura 26: Creació del marcador

- El constructor google.maps.Maker pren per valor tots els literals de l'objecte Marker options, el qual especifica les propietats inicials del marcador.
- El paràmetre position, indica la ubicació on mostrar el marcador.
- El paràmetre map, especifica el mapa de treball sobre el qual s'afegirà el marcador.

- El paràmetre title, indica el nom que es mostrarà a l'usuari quan aquest es seleccioni.

3.2.5.2 InfoWindow

Aquest genera el núvol que conté tota la informació referent al restaurant. Fa referència a la classe google.maps.InfoWindow, on la propietat més important és content. Mitjançant aquesta propietat referència el text contentString.

Finalment, per generar un "advert" cada cop que l'usuari pitgi sobre la icona del marcador, emprarem el mètode google.naps.event.addListener(). Els paràmetres referenciats són els següents:

- Marker: L'objecte a afegir
- El event al que ha de respondre, click en el nostre cas.
- La funció que s'executarà quan aquest es requereixi.

```
<!--Variable asociada al reacuadro de informaci[illegible]n del marker-->

    var infowindow = new google.maps.InfoWindow({
        content: contentString
    });


<!--Mostramos automaticamente el texto del marker-->

infowindow.open(map,marker);


<!--Referenciamos las variables a mostrar en el mapa-->

    google.maps.event.addListener(marker, 'click', function() {
        infowindow.open(map,marker);
    });
```

Figura 27: Diàleg de l'InfoWindow

3.2.5.3 Polilínees

Les polilínees son una sèrie de segments connectats entre si mitjançant vèrtexs i una seqüència ordenada. En la present aplicació, aquest s'ha emprat en la realització de la traça de ruta, mitjançant l'objecte DirectionsService.

```
function calcRoute() {

    var start = document.getElementById("routeStart").value;

        var request = {
            origin:start,
```

```
        destination: "ciutat de lleida 5 pont de suert, ES",
        travelMode: google.maps.DirectionsTravelMode.DRIVING,
        unitSystem: google.maps.UnitSystem.METRIC
    };

    directionsService.route(request, function(response, status) {


    <!--Si todo sale bi\xE9n, mostramos la ruta; en caso contrario

    if (status == google.maps.DirectionsStatus.OK) {
            directionsDisplay.setDirections(response);
        }
```

Figura 28: Càlcul de ruta a traçar

A continuació es descriuen els objectes referenciats a la variable request:

- Origin: Indica la posició introduïda per l'usuari en el formulari de cerca.
- Destination: Indica la posició del restaurant.
- TravelMode: Indica la configuració de com es realitzarà la traça de ruta. En la present aplicació s'emprarà Driving (conducció) com a medi de transport.
- UnitSystem: Indica la nomenclatura del sistema mètric que s'emprarà en la traça de ruta.
- Status: Indica si tot el procés anterior és correcte.

Finalment, si tot és correcte, s'encapsularà la resposta del servidor a l'interior de l'objecte DirectionsDisplay.

3.2.6 Control d'errors

Com s'ha presentat anteriorment, l'objecte status encapsula la resposta del servidor. En cas que aquesta sigui d'error, és aconsellable indicar a l'usuari el motiu de la fallida.

```
if (status == google.maps.DirectionsStatus.OK) {
        directionsDisplay.setDirections(response);
    }

else {
    if (status == 'ZERO_RESULTS') {
        alert('No se encontr\u00f3 ninguna ruta entre La Paradeta y tu direcci\u00f3n');
    } else if (status == 'UNKNOWN_ERROR') {
        alert('Contenido no disponible actualmente. Int\u00e9ntalo de nuevo m\u00e1s tarde
    } else if (status == 'REQUEST_DENIED') {
        alert('Contenido no disponible actualmente. Int\u00e9ntalo de nuevo m\u00e1s tarde
    } else if (status == 'OVER_QUERY_LIMIT') {
        alert('Contenido no disponible actualmente. Int\u00e9ntalo de nuevo m\u00e1s tarde
    } else if (status == 'NOT_FOUND') {
        alert('La direcci\u00f3n introducida no es v\u00e1lida o es incorrecta');
    } else if (status == 'INVALID_REQUEST') {
        alert('Nuestra direcci\u00f3n no se encuentra disponible temporalmente.');
```

```
    } else {
        alert("Error desconocido. C\u00f3digo de error: \n\n"+status);
    }
}


});
```

Figura 29: Control d'errors en càlcul de ruta

Com podem observar en la figura 29, en cas que l'objecte status retorni un valor diferent a google.maps.DirectionsStatus.OK, podem afirmar que s'ha produït una fallida. En la successió d'ifs s'han controlat tots els casos d'errors possibles, amb el corresponent *alert* per mostrar-lo a l'usuari.

4 Disseny tècnic

El present capítol especificarà el disseny tècnic, el qual desenvoluparà una explicació detallada de la confecció e implementació de la tecnologia presentada amb la finalitat de satisfer les necessitats requerides pel client.

4.1 Anàlisis de requeriments

L'anàlisi de requeriments és la base de tota aplicació, realitzant un símil amb el sector de la construcció, es podria anomenar com els ciments d'una casa. Els requeriments són els condicionadors del disseny i la implementació a realitzar.

En els capítols presentats fins al moment, s'ha introduït una breu menció als requeriments que l'aplicació havia de satisfer. L'anàlisi, segueix tot un recorregut anomenat procés tecnològic, on a la finalització de l'aplicació, s'ha de sotmetre-la a un estricte control de qualitat per tal de determinar el bon funcionament d'aquesta. El factor més rellevant de l'anàlisi, es realitzar una identificació completa del problema, on s'ha de remarcar clarament els objectius d'aquesta, la metodologia de treball i la lògica de negoci de l'aplicació.

La metodologia de treball, no ha estat la convencional en l'enginyeria de software de la programació web, a conseqüència que s'ha seguit la filosofia de treball d'un framework.

4.1.1 Requeriments funcionals

Els requeriments funcionals són aquells que especifiquen la lògica de sistema. Són els que determinen el comportament i pragmatisme de l'aplicació, ja que satisfan les necessitats que l'aplicació requereix. A continuació es detalla un breu esquema dels requeriments funcionals necessaris per confeccionar l'aplicació:

- Permetre la creació d'entitats les quals representin articles alimentaris.
- Permetre la creació d'entitats les quals representin usuaris.
- Permetre la creació d'entitats les quals representin reserves.
- Permetre la creació d'entitats les quals representin vendes.
- Assignació dels rols anònim, d'usuari i d'administrador.
- Edició, gestió i visualització d'usuaris, articles i reserves.

En les següents subseccions es detallen l'estructuració i el disseny dels requeriments funcionals.

4.1.1.1 Actors

Són els encarregats de realitzar la interacció entre les diferents parts que conformen l'aplicació. Estan conformats segons els tres rols definits prèviament. Trobem quatre tipus diferents:

- **Usuari anònim:** aquest actor estarà destinat a la visualització de continguts públics de l'aplicació.
- **Usuari autenticat:** aquest actor estarà destinat a la visualització e interacció dels continguts destinats en la zona privada de l'aplicació.
- **Administrador:** aquest actor estarà destinat a la gestió dels continguts que conformen l'àrea d'accés restringit de l'aplicació.
- **Symfony2:** aquest actor estarà destinat a la realització d'esdeveniments automàtics (encaminament, generació de bundles i entitats, generació de les plantilles que conformen la vista, ...).

4.1.1.2 Casos d'ús

La finalitat de l'elaboració dels casos d'us és detallar les accions que poden realitzar els actors definits anteriorment.

4.1.1.2.1 Usuari anònim

L'usuari anònim, podrà realitzar les següents accions:

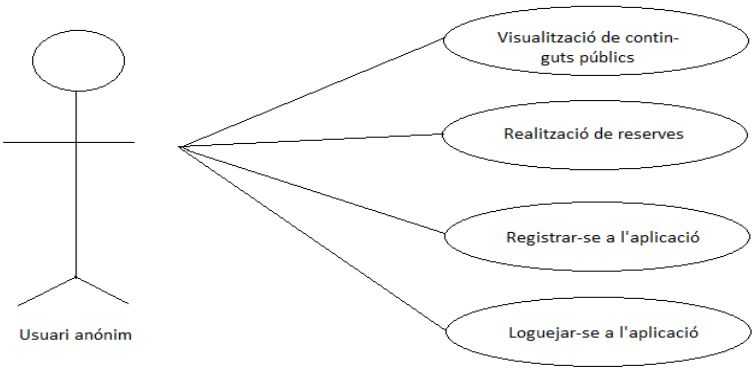


Figura 30: Cas d'ús usuari anònim

- Visualització dels continguts públics: Són aquells destinats a remetre informació sobre el restaurant (portada, Sidreria&Vinacoteca i altres continguts del menú de navegació).
    - La fitxa de l'article contindrà les següents dades:
        - Nom de l'article.
        - Descripció de l'article.
        - Preu de l'article.
        - Fotografia de l'article.
        - Botó de compra.
- Realització de reserves: Tot i que pot semblar un contingut pertanyent a la zona privada, no es així a causa que l'aplicació genera un formulari on l'usuari ha de remetre les seves dades. Com a contrapartida, l'aplicació auto completarà les dades de l'usuari autenticat.
    - El formulari de reserva requerirà les següents dades:
        - Nom.
        - Cognoms.
        - Correu electrònic.
        - Telèfon de contacte.
        - DNI.
        - Data de reserva.
        - Zona horària de reserva (dinar o sopar).
        - Total de comensals.
        - Suggeriments al cheff.
- Registre en l'aplicació: L'usuari inserirà les seves dades per tal d'accedir a la zona privada.
    - El formulari de registre requerirà les següents dades:
        - Nom.
        - Cognoms.

- Correu electrònic.
- Contrasenya (es realitzarà un procés de codificació).
- Telèfon de contacte.
- Direcció i codi postal.
- Data de naixement.
- DNI.

- L'aplicació generarà automàticament els següents continguts:
  - Salt: Cadena de caràcters aleatòria emprada per xifrar la contrasenya.
  - Contrasenya xifrada: Es codificarà la contrasenya mitjançant l'algoritme SHA512 amb un total de 10 iteracions, i el salt. Posteriorment serà emmagatzemada en la base de dades.

➢ Login a l'aplicació: L'usuari inserirà el seu correu i contrasenya per tal d'accedir a la zona privada.

4.1.1.2.2 Usuari autenticat

L'usuari autenticat, podrà realitzar les següents accions:

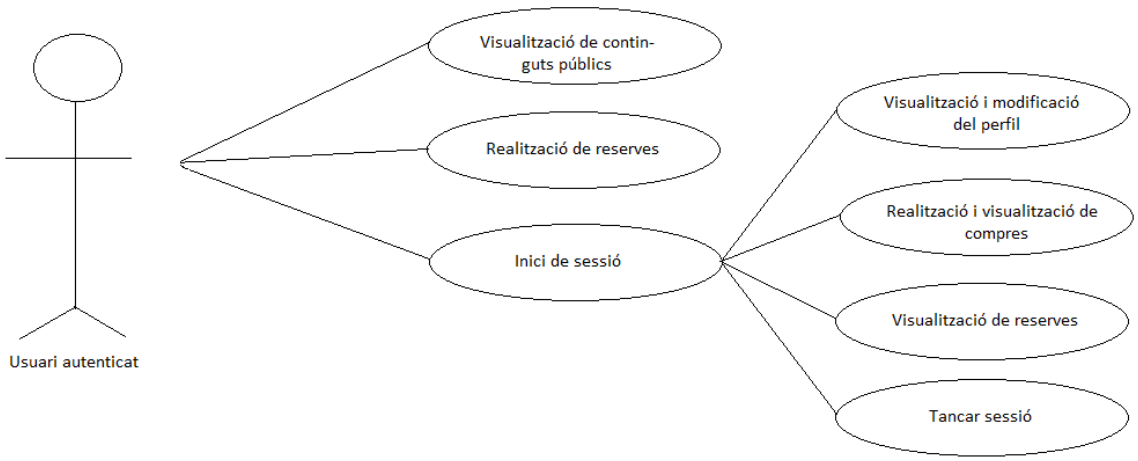


Figura 31: Cas d'ús usuari autenticat

➢ Inici de sessió: L'usuari s'autenticarà mitjançant el seu correu i contrasenya per tal d'accedir a la zona privada. Aquet posseirà l'opció de seleccionar un "checkbox" de no tancament de sessió, el qual emmagatzemarà la seva

"cookie" durant una setmana, amb la qual cosa no se l'hi requerirà les seves credencials durant el transcurs d'aquest període de temps.

- Visualització i modificació del perfil: L'usuari podrà consultar les dades facilitades en el procés de registre i modificar-les.
    - Dades del perfil:
        - Nom.
        - Cognoms.
        - Contrasenya (si nos es modifica, es manté l'actual).
        - Telèfon de contacte.
        - Direcció completa i codi postal.
        - Data de naixement.
        - DNI.
- Realització i visualització de compres: L'usuari podrà comprar els articles presentats en l'aplicació i consultar aquestos. Les dades que es requeriran i es visualitzaran són les següents:
    - Procés de compra:
        - Visualització de la fitxa de l'article a adquirir.
        - Data de recollida de l'article.
        - Zona horària de recollida.
        - Total d'articles a adquirir.
    - Visualització de compres realitzades:
        - Nom de l'article adquirit (es visualitzarà la seva fitxa mitjançant un hipervincle).
        - Data d'operació.
        - Data de recollida.
        - Zona horària de recollida.
        - Total d'unitats de l'article adquirit.
        - Import total.

- Visualització de reserves: L'usuari podrà consultar les reserves realitzades. Com que l'aplicació no emmagatzema l'id de l'usuari en la taula de la base de dades, s'emprarà el correu electrònic com a filtre. Les dades que es visualitzaran són les següents:
    - Visualització de reserves:
        - Data d'operació.
        - Data de reserva.
        - Zona horària de reserva.
        - Comensals.
        - Suggeriments al cheff.
- Finalització de sessió: L'usuari podrà tancar la sessió en qualsevol instant de temps, on aquest tornarà a ser un actor anònim.

4.1.1.2.3 Usuari administrador

L'usuari administrador, podrà realitzar les següents accions:

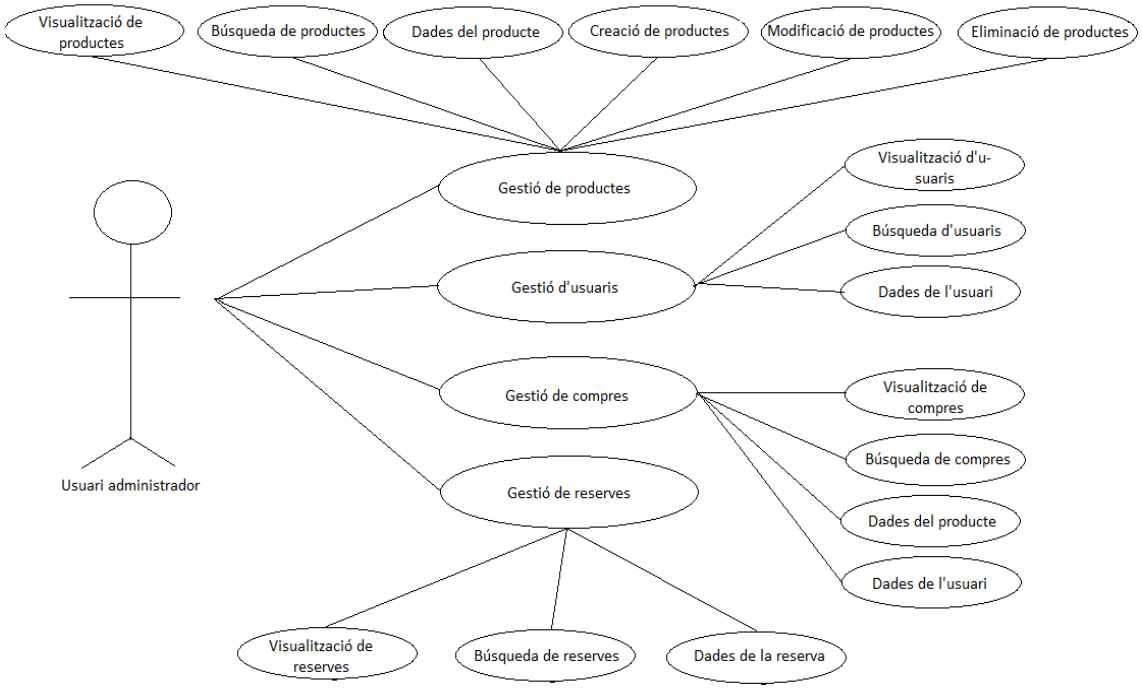


Figura 32: Cas d'ús usuari administrador

- Gestió de productes: L'administrador podrà realitzar les següents accions:

- Visualització de productes: Es mostrarà la fitxa de l'article en una composició estructurada en forma de taula. Les dades que es visualitzaran són les següents:
  - Nom.
  - Descripció.
  - Preu.
  - Imatge (es visualitzarà el nom de la imatge).
  - Modificació (es redireccionarà al formulari de modificació mitjançant un hipervincle).
  - Eliminació (es redireccionarà al formulari d'eliminació mitjançant un hipervincle).
- Búsqueda de productes: Es mostrarà la un formulari de búsqueda per nom d'article.
- Dades del producte: Es mostrarà la fitxa de l'article. Les dades que es visualitzaran són les següents:
  - Nom.
  - Descripció.
  - Preu.
  - Imatge (es visualitzarà la fotografia).
  - Modificació (es redireccionarà al formulari de modificació mitjançant un hipervincle).
  - Eliminació (es redireccionarà al formulari d'eliminació mitjançant un hipervincle).
- Creació de productes: Es mostrarà un formulari de creació d'articles. Les dades que es requeriran són les següents:
  - Nom.
  - Descripció.
  - Preu.
  - Imatge (si no es selecciona cap, s'emmagatzemarà la ruta de la imatge per defecte).

        - Tipologia o família d'articles a la qual pertany. Trobem les següents classes:
            - Sidra.
            - Vins.
            - Caves.
            - Tapes.
            - Peix.
            - Entrants.
            - Carns.
    - Modificació de productes: Es mostrarà un formulari de modificació d'articles. Les dades que, si escau, es modificaran són les següents:
        - Nom.
        - Descripció.
        - Preu.
        - Imatge (si no es selecciona cap, s'emmagatzemarà la ruta de la imatge introduïda anteriorment en la base de dades).
        - Tipologia.
    - Eliminació de productes: S'eliminarà el producte seleccionat.
- Gestió d'usuaris: L'administrador podrà realitzar les següents accions:
    - Visualització d'usuaris: Es mostrarà la fitxa de l'usuari en una composició estructurada en forma de taula. Les dades que es visualitzaran són les següents:
        - Correu electrònic (es redireccionarà a la seva fitxa completa mitjançant un hipervincle).
        - Nom.
        - Cognoms.
        - Telèfon de contacte.
        - Data d'alta a l'aplicació.

- Búsqueda d'usuaris: Es mostrarà la un formulari de búsqueda per nom d'usuari.
- Dades de l'usuari: Es mostrarà la fitxa de l'usuari. Les dades que es visualitzaran són les següents:
  - Nom.
  - Cognoms.
  - Correu electrònic.
  - DNI.
  - Telèfon de contacte.
  - Direcció i codi postal.
  - Data d'alta a l'aplicació.
  - Aniversari.

- Gestió de vendes: L'administrador podrà realitzar les següents accions:
  - Visualització de compres: Es mostrarà les compres vigents a dia d'avui en una composició estructurada en forma de taula. Les dades que es visualitzaran són les següents:
    - Nom del producte (es redireccionarà a la seva fitxa de l'article mitjançant un hipervincle).
    - Correu electrònic (es redireccionarà a la seva fitxa de l'usuari mitjançant un hipervincle).
    - Nom de l'usuari.
    - Telèfon de contacte.
    - Data de recollida.
    - Zona horària de recollida.
    - Total d'articles adquirits.
  - Búsqueda de vendes: Es mostrarà un formulari de búsqueda per data de compra.
  - Dades de l'usuari: Es mostrarà la fitxa de l'usuari mitjançant un hipervincle prèviament configurat.

- Dades del producte: Es mostrarà la fitxa del producte mitjançant un hipervincle prèviament configurat.

- Gestió de reserves: L'administrador podrà realitzar les següents accions:
  - Visualització de reserves: Es mostrarà les reserves vigents a dia d'avui en una composició estructurada en forma de taula. Les dades que es visualitzaran són les següents:
    - Correu electrònic (es redireccionarà a la seva fitxa mitjançant un hipervincle).
    - Nom de l'usuari.
    - Telèfon de contacte.
    - Data de reserva.
    - Zona horària de reserva.
    - Comensals.
    - Suggeriments al cheff.
  - Búsqueda de reserves: Es mostrarà un formulari de búsqueda per data de reserva.
  - Dades de l'usuari: Es mostrarà la fitxa de l'usuari. Les dades que es visualitzaran són les següents:
    - Nom.
    - Cognoms.
    - Correu electrònic.
    - Telèfon de contacte.
    - DNI.
    - Data d'operació.
    - Data de reserva.
    - Zona horària de reserva.
    - Comensals.
    - Suggeriments al cheff.

4.1.1.3 Restriccions de la lògica de negoci

Segons l'especificació esmentada pel client, l'aplicació ha de ser capaç de processar les següents restriccions:

- Gestió de reserves:
  - L'hora màxima per realitzar reserves per dinar vigents a dia d'avui és les 13:00.
  - L'hora màxima per realitzar reserves per sopar vigents a dia d'avui és les 20:00.
  - L'aforament màxim destinat a reserves gestionades mitjançant l'aplicatiu web és de 50 comensals.
- Gestió de compres:
  - L'hora màxima per realitzar compres per dinar vigents a dia d'avui és les 13:00.
  - L'hora màxima per realitzar compres per sopar vigents a dia d'avui és les 20:00.
  - La varietat de marisc i peixos tant sols es troba disponible els caps de setmana (compres per recollida el dissabte o diumenge).
  - S'ha estimat un estocatge d'articles il·limitat, degut a especificació explícita del client.

4.1.2 Requeriments no funcionals

Els requeriments no funcionals són aquells que no formen part implícita de l'aplicació. Tot i no formar part de la lògica de negoci, la seva estructuració és vital a causa que d'ells depèn l'èxit de l'aplicació. A continuació es detallen els requeriments no funcionals necessaris per confeccionar l'aplicació:

- Multi-plataforma: L'aplicació s'havia de poder executar en la gran veritat de navegadors els quals es troben a disposició de l'usuari.
- Interfície intuïtiva: L'aplicació ha de ser accessible a tot tipus d'usuaris, on els elements d'aquesta han de ser esclaridors, ben estructurats i les interaccions amb aquesta no han de requerir alts coneixements d'informàtica.
- Interfície atractiva: L'aplicació ha d'estar adaptada als nous corrents i ten-

dències de disseny web, on el seu aspecte no ha de donar la sensació de desfasada.

- Accessible: L'aplicació ha de permetre accedir als diferents entorns i serves el més ràpid possible, i amb el menor nombre d'enllaços possible.
- Manteniment: El manteniment de l'aplicació en la fase de producció hauria de ser mínim, tots els continguts han de poder ser generats per l'administrador sense la necessitat de la intervenció del programador.
- Extensibilitat: L'aplicació ha de permetre generar un ampli catàleg de productes sense la necessitat de modificar el codi existent.
- Seguretat: L'aplicació ha de blindar les dades facilitades pels usuaris contra els possibles atacs que es puguin rebre, i garantir l'autenticació quan aquesta es requereixi.
- Disponibilitat: Els continguts presentats en l'aplicació, han d'estar disponibles en qualsevol moment.

4.2 Disseny del Frontend

En aquesta secció es presentarà informació sobre l'elaboració de la zona d'usuaris de l'aplicació.

4.2.1 Model de base de dades

Definits els requeriments funcionals, es va optar per la creació de quatre taules emprant un model de base de dades relacional, per tal de satisfer les necessitats especificades en els requisits preliminars de l'aplicació:

- Producte, la finalitat de la qual és emmagatzemar informació referent als articles disponibles.
- Venda, la finalitat de la qual és emmagatzemar informació referent a les vendes realitzades pels usuaris.
- Reserva, la finalitat de la qual és emmagatzemar informació referent a les reserves realitzades pels usuaris.
- Usuari, la finalitat de la qual és emmagatzemar informació referent als usuaris.

Les relacions entre les taules són les següents:

- Venda – Producte: 1-n, una venda sol pot pertànyer a 1 producte, però un producte pot estar associat a n vendes realitzades.
- Venda – Usuari: 1-n, una venda sol pot pertànyer a 1 usuari, però un usuari pot realitzar n compres.

Els camps que aquestos han de contenir són els següents:

- Taula *Producte*

Columna	Tipus	Comentaris
id	int	Clau primària, auto-incrementable
nom	Varchar (255)	
descripció	Varchar (255)	
classe	Varchar (255)	Indicarà a quina família pertany
fotografia	Varchar (255)	Sol emmagatzema la ruta de la foto
preu	Decimal (10,2)	En euros

Taula 7: Taula Producte

- Taula *Venda*

Columna	Tipus	Comentaris
id	int	Clau primària, auto-incrementable
producte_id	int	Relació 1:n, clau forana
usuari_id	int	Relació 1:n, clau forana
data_operació	datetime	
fdata_recollida	date	
tipus	Varchar (255)	d en cas de dinar; c en cas de sopar
quantitat	int	Total d'articles adquirits

Taula 8: Taula Venda

- Taula *Reserva*

Columna	Tipus	Comentaris
id	int	Clau primària, auto-incrementable
nom	Varchar (255)	
cognoms	Varchar (255)	
email	Varchar (255)	
telèfon	Varchar (9)	
DNI	Varchar (9)	
comensals	int	
data_operació	datetime	
tipus	Varchar (255)	d en cas de dinar; c en cas de sopar
data_reserva	date	

Comentaris	longtext	Suggeriments al cheff

Taula 9: Taula Reserva

El motiu de no emmagatzemar l'identificador de l'usuari, és degut a que no és necessari registrar-se per tal de realitzar una reserva. Com a contrapartida, per al mostreig de les reserves dels usuaris registrats, s'emprarà l'email com a filtre.

- Taula *Usuari*

Columna	Tipus	Comentaris
id	int	Clau primària, auto-incrementable
nom	Varchar (255)	
cognoms	Varchar (255)	
email	Varchar (255)	Emprat també com a *login*
password	Varchar (255)	
salt	Varchar (255)	Valor aleatori emprat per xifrar la contrasenya
telèfon	Varchar (9)	
DNI	Varchar (9)	
direcció	longtext	
data_alta	datetime	
data_naixement	date	

Taula 10: Taula Usuari

4.2.2 Generació de Bundles

Es generarà tres bundles, seguint la lògica definida en el model relacional de base de dades.

- ProductoBundle, formarà part del directori públic del frontend.
- UsuarioBundle, formarà part del directori privat del frontend (gestió de les accions de l'usuari).
- BackendBundle, formarà part del directori d'accés restringit de l'aplicació (gestió de l'administrador).

Symfony2, proveeix al programador d'un servei el qual genera l'*esquelet* complert d'un *bundle*. Per a fer ús d'aquest, executarem la següent instrucció:

```
$ php app/console generate:bundle
```

Figura 33: Servei de generació d'un bundle

Aquest, formularà cinc qüestions:

- *Bundle namespace*, és el *namespace* amb el que es crearà el bundle, conseqüentment, determina el directori on aquest s'emmagatzema. Seguint la nomenclatura estàndard, es composarà de dos parts:
  - Nom del projecte amb el qual es desenvolupa el bundle, Paradeta en el nostre cas.
  - Nom del propi bundle, el qual sempre ha d'acabar amb la terminació Bundle.

  Finalment, la resposta a la primera qüestió seria:
  - Paradeta/ProductoBundle
  - Paradeta/UsuarioBundle
  - Paradeta/BackendBundle
- *Bundle name*, és el nom el qual referència el *bundle*. L'estàndard recomana que aquest sigui simplement el *namespace*. La resposta a la segona qüestió seria:
  - ProductoBundle
  - UsuarioBundle
  - BackendBundle
- *Target directory*, directori en el qual s'emmagatzemaran els *bundles*. La resposta a la tercera qüestió seria:
  - Src/
- *Configurate format*, determina el format emprat en els arxius de configuració del *bundle*. La resposta a la quarta qüestió seria:
  - yml
- *Do you want to generate the whole diretory structure?*, genera una estructura personalitzada de directoris. La resposta a la cinquena qüestió seria:
  - no, com a conseqüència es generarà una estructura estàndard.

Finalment, aquest requerirà dos últimes qüestions:

- *Confirm automatic update of your Kernel?*, activarà el bundle per tal d'estar disponible. La resposta a la quarta qüestió seria:

- yes

➢ *Confirm automatic update of your Routing?*, importarà la configuració d'encaminament del *bundle* des de l'arxiu general de l'aplicació. La resposta a la quarta qüestió seria:

- yes

A la finalització del qüestionari, es realitzaran els següents canvis automàtics:

➢ Es generarà la següent estructura d'arxius i directoris a l'interior del *bundle*.

```
ProductoBundle.php
Controller/
  DefaultController.php
DependencyInjection/
  Configuration.php
  ProductoExtension.php
Resources/
  config/
    routing.yml
    services.yml
  views/
    Default/
      index.html.twig
```

Figura 34: Jerarquia de directoris d'un bundle

➢ S'activarà el nou *bundle*, el qual quedarà enregistrat a l'interior de la classe AppKernel.php, la qual es el nucli de l'aplicació. Es generarà la crida al *bundle* de la següent forma:

- new Paradeta\ProductoBundle\ProductoBundle()

4.2.3 Definició d'entitats

Definit el model relacional, es generaran les entitats per tal de traduir la informació de les taules de la base de dades a classes PHP:

➢ Entitat Producte, ubicada al *bundle* ProductoBundle

➢ Entitat Venda, ubicada al *bundle* ProductoBundle

➢ Entitat Reserva, ubicada al *bundle* ProductoBundle

➢ Entitat Usuari, ubicada al *bundle* UsuarioBundle

Symfony2, proveeix al programador d'un servei el qual genera automàticament entitats completes. Per a fer us d'aquest, executarem la següent sentencia:

```
$ php app/console doctrine:generate:entity

Welcome to the Doctrine2 entity generator

This command helps you generate Doctrine2 entities.
```

Figura 35: Servei de generació d'una entitat

Aquest, plantejarà cinc qüestions:

- *Entity shortcut name*, és el nom abreviat de l'entitat, el qual es forma concatenant el nom del *bundle* i el de la classe de l'entitat. La resposta a aquesta qüestió seria:
  - ProductoBundle:Producto
  - ProductoBundle:Reserva
  - ProductoBundle:Venta
  - UsuarioBundle:Usuario

  Aquesta acció, generarà l'entitat (per exemple Producto.php) a l'interior del *bundle*.

- Les anotacions són el format recomanat per a definir les entitats.

```
Determine the format to use for the mapping information.

Configuration format (yml, xml, php, or annotation) [annotation]: <Enter>
```

Figura 36: Definició d'anotacions a l'entitat

- Després d'indicar la informació bàsica, aquest sol·licita informació de totes les propietats:

```
New field name (press <return> to stop adding fields): nombre
Field type [string]: <Enter>
Field length [255]: <Enter>
```

Figura 37: Definició de les entitats

Primerament, s'indica el nom de la propietat (en aquest cas en particular nombre). Després, el tipus d'informació al que pertany aquesta (per defecte string), i finalment s'indicarà la longitud màxima de la cadena de text (per defecte 255).

No és necessari crear la propietat $id que actua com a clau primària perquè aquest la genera automàticament.

- L'inconvenient més gran que presenta, és la falta de flexibilitat, a causa que no permet referenciar una entitat com un tipus de clau forana. La modificació a realitzar és la següent:

```
/**
 * @ORM\ManyToOne(targetEntity="Paradeta\ProductoBundle\Entity\Producto")
 */
protected $producto;

/**
 * @ORM\ManyToOne(targetEntity="Paradeta\UsuarioBundle\Entity\Usuario")
 */
protected $usuario;
```

Figura 38: Definició de claus foranes

Per a l'especificació de claus foranes, s'emprarà la notació @ORM\ManyToOne(targetEntity="Paradeta\(bundle a seleccionar)Bundle\Entyty\(entitat a manipular), seguint la lògica definida en l'especificació del model relacional de dades.

4.2.3.1 Manipulació d'entitats

Doctrine2 proveeix al programador de mètodes especials per tal d'aconseguir modificar el valor d'aquestes. Aquets mètodes s'anomenen getter i setter.

El mètode getter s'empra per consultar el valor d'una entitat, mentre que el mètode setter s'empra per modificar el valor d'aquesta. L'estàndard de definició a implementar és el següent:

- Mètode getter:
  - Public function getNombre()

    {

    Return $this -> nombre;

    }

    Com podem observar, primer es defineix l'acció a realitzar i com a resposta d'aquesta, al seu interior es defineix el valor a retornar.
- Mètode setter:

- Public function setNombre($nombre)

  {

  $this -> nombre = $nombre;

  }

  Com podem observar, primer es defineix l'acció a realitzar, on es parametritza el nou valor de l'entitat, en aquest cas en particular $nombre.

  Com a resposta d'aquesta, al seu interior es defineix l'entitat a manipular ($this -> nombre) i el nou valor ($nombre).

Doctrine2, s'encarrega d'establir automàticament el valor dels setters i getters de les entitats. Però l'inconvenient sobre la flexibilitat presentat anteriorment no és una excepció, on la modificació de getters i setters de les claus foranes a realitzar és la següent:

```
/**
 * Set producto
 *
 * @param Paradeta\ProductoBundle\Entity\Producto $producto
 */
public function setProducto(\Paradeta\ProductoBundle\Entity\Producto $producto)
{
    $this->producto = $producto;
}

/**
 * Get producto
 *
 * @return Paradeta\ProdcutoBundle\Entity\Producto
 */
public function getProducto()
{
    return $this->producto;
}

/**
 * Set usuario
 *
 * @param Paradeta\UsuarioBundle\Entity\Usuario $usuario
 */
public function setUsuario(\Paradeta\UsuarioBundle\Entity\Usuario $usuario)
{
    $this->usuario = $usuario;
}

/**
 * Get usuario
 *
 * @return Paradeta\UsuarioBundle\Entity\Usuario
 */
```

```
public function getUsuario()
{
    return $this->usuario;
}
```

Figura 39: Manipulació de claus foranes

La notació @return Paradeta\(bundle a seleccionar)Bundle\Entyty\(entitat a obtenir) dels getters i la @param Paradeta\(bundle a seleccions)Bundle\(entitat a manipular) $valor dels setters configuren la manipulació de les claus foranes. El @return referència la ubicació i valor d'aquesta; en el cas del setter, el @param referència la ubicació i el valor a parametritzar.

4.2.3.2 Mètodes especials

Els mètodes especials són aquells que configuren la definició, conversió i manipulació automàtica de les entitats relacionals configurades anteriorment:

- Mètode de definició i construcció temporal:

```
/**
 * Constructor para generar automaticamente la fecha de compra
 */
public function __construct()
{
    $this->fecha_compra = new \DateTime();
}
```

Figura 40: Constructor de dates

Aquest constructor genera automàticament l'hora i data de compra de l'article.

- Mètode de conversió a cadena de text:

```
public function __toString()
{
    return $this->getNombre;
}
```

Figura 41: Conversor a cadena text

Aquest mètode defineix la conversió d'una entitat a una cadena de text. Aquest és molt útil a l'hora de treballar amb entitats relacionals, a causa que transforma un identificador en el nom d'un article o el d'un usuari.

4.2.3.3 Anotacions

Tècnicament, les anotacions són comentaris introduïts en el propi codi font de l'aplicació, les quals són interpretades com accions per Doctrine2. Les anotacions de Doc-

trine2 es creen mitjançant la classe Doctrine\ORM\Mapping. A continuació es detallen les emprades en l'aplicació:

- @ORM\Entity
  - Aquesta anotació comporta que Doctrine2 emmagatzemi les dades de l'objecte en una taula anomenada igual que la classe PHP a la qual pertany.
- @ORM\Table(name="nom de la taula")
  - Aquesta anotació comporta que Doctrine2 emmagatzemi les dades de l'objecte en la taula referenciada.
- @ORM\GeneratedValue(trategy="AUTO")
  - Aquesta anotació comporta que Doctrine2 interpreti la propietat id com una estructura auto incrementable i única.
- @var integer $propietat
  - Aquesta anotació comporta que Doctrine2 l'interpreti com una variable de tipus enter.
- @var string $propietat
  - Aquesta anotació comporta que Doctrine2 l'interpreti com una variable de tipus string.
- @var date $propietat
  - Aquesta anotació comporta que Doctrine2 l'interpreti com una variable de tipus data.
- @ORM\Column(name="nom de la propietat", type = "tipus", length = longitud)
  - Aquesta anotació realitza la mateixa acció que les presentades anteriorment, però definint una longitud màxima d'aquesta.

4.2.3.4 Asserts

Els asserts són *constraints* o regles de validació. La configuració estàndard, segueix la mateixa nomenclatura que la presentada fins al moment, mitjançant el prefixe @Assert\. Symfony2 inclou una llibreria amb nombroses *constraints*, a continuació es presenten les emprades en l'aplicació:

- Validacions bàsiques:
  - @Assert\NotBlank(), el valor no és null o una cadena buida de text.
  - @Assert\Type(), el valor és del tipus indicat en l'opció type. Per exemple:
    - @Assert\Type(type="int").
- Validacions per a cadenes de text:
  - @Assert\Email(), el valor ha de tenir l'aspecte d'un correu electrònic vàlid. Si es vol assegurar que aquest existeix, podem emprar l'opció checkMX, la qual empra la funció checkdnsrr() de PHP.
  - @Assert\MiniLength(limit= 6), el valor ha de tenir com a mínim el número de caràcters especificats en l'opció limit.
  - @Assert\MaxLength(limit= 6), el valor pot contindre el màxim de caràcters especificats en l'opció limit.
  - @Assert\Regex(), el valor ha de complir el patró de la expressió regular definida en l'opció pattern. Per exemple:
    - @Assert\Regex(pattern="/\d{9}/"), validació emprada per a números de telèfon de nou xifres.
- Validacions per a enters:
  - @Assert\Max(limit=30), el valor no pot ser superior al nombre indicat en l'opció limit.
  - @Assert\Min(limit=1), el valor no pot ser inferior al nombre indicat en l'opció limit.
- Validacions per a dates:
  - @Assert\Date(), el valor és un objecte de tipus Date o una cadena de text amb el format YYYY-MM-DD.
  - @Assert\DateTime(), el valor és un objecte de tipus DateTime o una cadena de text amb el format YYYY-MM-DD HH:MM:SS.
- Validacions per a col·leccions:
  - @Assert\Choice(), el valor es troba definit dins del rang de possibilitats especificats en l'opció choices. Per exemple:

- @Assert\Choice(choices = {"sidra","vins","cava","tapes","entrants","peixos","carns"}).

- @DoctrineAssert\UniqueEntity(), assegura que el valor és únic dins de la mateixa entitat, es a dir, el seu valor no es repeteix en altres files de la taula de la base de dades. Per exemple:

  - @ DoctrineAssert\UniqueEntity("email").

- Validacions d'arxius:

  - @Assert\File(), el valor és del tipus File, UploadFile o una cadena de text la qual correspon a la ruta d'un arxiu existent. Disposa de l'opció maxSize, per tal de limitar la seva mida en bytes.

- Validacions especials:

  Les validacions especials són aquelles les quals no estan definides dins de la llibreria de Symfony2 i es tasca del programador determinar la seva lògica.

  - @Assert\Callback(methods={"esDniValido"}), el valor es valida mitjançant el *callback* de PHP indicat en l'opció methods.

Quan es produeix un error de validació, Symfony2 retorna un missatge d'error predefinit pera cada *constraint*. També trobem disponible l'edició de missatges d'error personalitzats, mitjançant el paràmetre message:

```
/**
 * @Assert\NotBlank(message = "Por favor, escribe tu nombre")
 */
protected $nombre;
```

Figura 42: Missatge d'error personalitzat

4.2.3.4 Regles de validació pròpies

Tot i que les *constraints* incloses en la llibreria de Symfony2 inclouen les validacions més rellevants, s'ha implementat una nova lògica per tal de verificar l'autenticitat del DNI introduït per l'usuari.

```
public function esDniValido(ExecutionContext $context)
{
    $nombre_propiedad = $context->getPropertyPath() . '.dni';
    $dni = $this->getDni();

    // Comprobar que el formato sea correcto
```

```
    if (0 === preg_match("/\d{1,8}[a-z]/i", $dni)) {
        $context->setPropertyPath($nombre_propiedad);
        $context->addViolation('El DNI introducido no tiene el formato correcto (entre 1
        y 8 números seguidos de una letra, sin guiones y sin dejar ningún espacio en bla
        nco)', array(), null);
        return;
    }

    // Comprobar que la letra cumple con el algoritmo
    $numero = substr($dni, 0, -1);
    $letra  = strtoupper(substr($dni, -1));
    if ($letra != substr("TRWAGMYFPDXBNJZSQVHLCKE", strtr($numero, "XYZ", "012")%23, 1)) {
        $context->setPropertyPath($nombre_propiedad);
        $context->addViolation('La letra no coincide con el número del DNI. Comprueba que
        has escrito bien tanto el número como la letra', array(),null);

    }
}
```

Figura 43: Funció de validació del DNI

Els mètodes de validació personalitzats no retornen cap valor, ni quan la validació és errònia. El seu únic propòsit és afegir *violacions* de validació quan aquesta es produeixi. Les *violacions* emprem l'objecte de tipus ExecutionContext, el qual és parametritzat automàticament per Symfony2 com argument del mètode propi de validació.

En el cas específic del DNI, es poden produir dos tipus de *violacions*:

- El format no sigui l'adequat (entre un i vuit números seguits d'una lletra).
- La lletra no correspongui al número introduït.

Les *violacions* es comproven en els següents passos:

- S'obté el *nom complet* de la propietat concatenant el valor retornat pel mètode getPropertyPath() i la propietat $context->getPropertyPath() . '.dni'
- S'empra el mètode setPropertyPath() per tal d'indicar la propietat a la qual se l'hi afegeix la *violació*. Se l'hi parametritza el *nom complet* de la propietat obtingut anteriorment ($nombre_propiedad).
- Es crea la *violació* mitjançant el mètode addViolation(). El primer argument d'aquesta funció es el missatge d'error a mostrar a l'usuari, el segon es un array opcional de paràmetres i el tercer es pot emprar per retornar el codi d'error que ha produït la violació.

4.2.3.4.1 Mètodes de validació ad-hoc

Symfony2 també inclou un estàndard de definició de validadors personalitzats els quals no es troben inclosos en el ventall exposat dels *constraints*, amb la limitació que

el missatge d'error és únic per a cada funció. El seu funcionament es basa en la creació d'un mètode anomenat isXXX() a l'interior de l'entitat i associar-li una regla de validació de tipus @Assert\True.

```
/**
 * @Assert\True(message = "Debes tener al menos 18 años para registrarte en el sitio")
 */
public function isMayorDeEdad()
{
    return $this->fecha_nacimiento <= new \DateTime('today - 18 years');
}
```

Figura 44: Mètode de validació ad-hoc

Les dades d'una entitat sols es consideren vàlids si acompleixen totes les regles de validació definides en la funció.

4.2.4 Definició de l'encaminament

S'ha emprat la següent metodologia d'encaminament, seguint la lògica de casos d'us.

- Direccions associades a la portada:
  - _portada:
    - pattern: /
    - defaults: {_controller:ProductoBundle:Default:portada}

    El pattern: / indica que aquesta és la direcció per defecte de l'aplicació.
  - _portada:
    - pattern: /portada
    - defaults: { _controller: ProductoBundle:Default:portada}
- Direccions associades al mostregi d'articles:
  - sidra:
    - pattern: /sidra
    - defaults: {_controller:ProductoBundle:Default:sidra}
  - menu:
    - pattern: /menu

        - defaults: {_controller:ProductoBundle:Default:menu}
    - tapas:
        - pattern: /tapas
        - defaults: {_controller:ProductoBundle:Default:tapas}
    - pescados:
        - pattern: /pescados
        - defaults: {_controller:ProductoBundle:Default:pescados}

- Direcció associada a la realització de reserves:
    - reserva:
        - pattern: /reserva
        - defaults: {_controller: ProductoBundle:Default:reserva}
- Direcció associada a la localització del restaurant:
    - localizacion:
        - pattern: /localizacion
        - defaults: {_controller:ProductoBundle:Default:localizacion}
- Direccions associades a les pàgines estàtiques:
    - contacto:
        - pattern: /contacto
        - defaults: {_controller: ProductoBundle:Default:contacto}
    - aviso_legal:
        - pattern: /aviso_legal
        - defaults: {_controller:ProductoBundle:Default:aviso}
    - condiciones:
        - pattern: /condiciones
        - defaults: {_controller:ProductoBundle:Default:condiciones}
- Direccions associades als missatges d'error:

- error1:
  - pattern: /error1
  - defaults: {_controller:ProductoBundle:Default:error1}

  Error produït a causa de superar l'hora límit de reserves per dinar vigents a dia d'avui i hora major a les 13:00.

- error2:
  - pattern: /error2
  - defaults: {_controller:ProductoBundle:Default:error2}

  Error produït a causa de superar l'hora límit de reserves per sopar vigents a dia d'avui i hora major a les 20:00.

- error3:
  - pattern: /error3
  - defaults: {_controller:ProductoBundle:Default:error3}

  Error produït a causa de superar l'aforament màxim (50 comensals) destinat per reserves realitzades mitjançant l'aplicació web.

- error4:
  - pattern: /error4
  - defaults: {_controller:ProductoBundle:Default:error4}

    Error com a conseqüència d'haver seleccionat l'id d'un article inexistent en la base de dades.

➢ Direccions associades als missatges de confirmació:

- confirmacion5:
  - pattern: /confirmacion5
  - defaults: {_controller:ProductoBundle:Default:confirmacion5}

    Reserva per dinar completada satisfactòriament.

- confirmacion6:
  - pattern: /confirmacion6

- defaults: {_controller:ProductoBundle:Default:confirmacion6}

  Reserva per sopar completada satisfactòriament.

- confirmacion7:
  - pattern: /confirmacion7
  - defaults: {_controller:ProductoBundle:Default:confirmacion7}

    Missatge enviat satisfactòriament.

- Direcció associada al login d'usuaris:
  - usuario_login:
    - pattern: /usuario/login
    - defaults: {_controller:UsuarioBundle:Default:login}
- Direcció associada al registre d'usuaris:
  - usuario_registro:
    - pattern: /usuario/registro
    - defaults: {_controller:UsuarioBundle:Default:registro}
- Direcció associada al perfil de l'usuari:
  - usuario_perfil:
    - pattern: /usuario/perfil
    - defaults: {_controller:UsuarioBundle:Default:perfil}
- Direcció associada a l'acció de comprar:
  - usuario_comprar:
    - pattern: usuario/comprar/{alimento}
    - defaults: {_controller:UsuarioBundle:Default:comprar}
- Direcció associada a la visualització de compres realitzades:
  - usuario_compras:
    - pattern: /usuario/compras

        - defaults: {_controller:UsuarioBundle:Default:compras}

- Direcció associada a la fitxa de l'article adquirit:
    - adquirido:
        - pattern: adquirido/{alimento}
        - defaults: {_controller ProductoBundle:Default:adquirido}
- Direcció associada a la visualització de reserves realitzades:
    - usuario_reservas:
        - pattern: /usuario/reservas
        - defaults: {_controller:UsuarioBundle:Default:reservas}
- Direcció associada a la verificació de les credencials de l'usuari:
    - usuario_login_check:
        - pattern: /usuario/login_check
- Direcció associada a la desconnexió de l'usuari:
    - usuario_logout:
        - pattern: /usuario/logout
- Direccions associades als missatges de confirmació:
    - usuario_confirmacion1:
        - pattern: /usuario/confirmacion1
        - defaults: {_controller:UsuarioBundle:Default:confirmacion1}

          Registre completat satisfactòriament.
    - usuario_confirmacion2:
        - pattern: /usuario/confirmacion2
        - defaults: {_controller:UsuarioBundle:Default:confirmacion2}

          Perfil modificat satisfactòriament.
    - usuario_confirmacion3:
        - pattern: /usuario/confirmacion3

- defaults: {_controller:UsuarioBundle:Default:confirmacion3}

  Compra per dinar completada satisfactòriament.

- usuario_confirmacion4:
  - pattern: /usuario/confirmacion4
  - defaults: {_controller:UsuarioBundle:Default:confirmacion4}

    Compra per sopar completada satisfactòriament.

- Direccions associades als missatges d'error:
  - usuario_error5:
    - pattern: /usuario/error5
    - defaults: {_controller:UsuarioBundle:Default:error5}

      Error com a conseqüència d'haver seleccionat la recollida del sortit de peix i marisc un dia de la setmana diferent al cap de setmana.
  - usuario_error6:
    - pattern: /usuario/error6
    - defaults: {_controller:UsuarioBundle:Default:error6}

      Error produït a causa de superar l'hora límit de compres per dinar vigents a dia d'avui i hora major a les 13:00
  - usuario_error7:
    - pattern: /usuario/error7
    - defaults: {_controller:UsuarioBundle:Default:error7}

      Error produït a causa de superar l'hora límit de compres per sopar vigents a dia d'avui i hora major a les 20:00

4.2.4.3 Encaminament dinàmic

L'encaminament dinàmic és aquell el qual és capaç de suportar la parametrització d'un valor. Observem la següent direcció lògica:

- usuario_comprar:

pattern: usuario/comprar/{alimento}

defaults: {_controller: UsuarioBundle:Default:comprar}

El paràmetre {alimento} referenciat en la direcció, determina l'id del article a adquirir, el qual és emprat en el controlador per tal de processar la compra.

4.2.4.4 Interpretació de les direccions d'encaminament

Symfony2 interpreta les direccions d'encaminament com:

- Quan l'usuari sol·liciti una *URL* amb l'aspecte /portada, executa el mètode portadaAction() de la classe src/Paradeta/ProductoBundle/controller/DefaultController.php.

```
public function portadaAction()
{
    return $this->render('ProductoBundle:Default:portada.html.twig');
}
```

Figura 45: Acció a la sol·licitud de la portada

El controlador interpreta la funció mitjançant la *notació bundle*:

- ProductoBundle, és el nom del bundle on es troba la plantilla.
- Default, és el nom del directori on es troba ubicada la plantilla. Per defecte, Symfony2 identifica aquest a l'interior de Resources/views.
- Portada.html.twig, és el nom específic de la plantilla.

4.2.4 Plantilles twig

Twig és un motor i llenguatge de plantilles PHP molt ràpid i eficient, emprat en el procés de generació de la vista. Twig empra infinitat d'etiquetes, per tal de delimitar la seva funcionalitat, les més rellevants es presenten en les següents subseccions.

4.2.4.1 Mostreig d'informació

Les pàgines HTML que s'enviaran a l'usuari, sovint es generen de forma dinàmica mitjançant les plantilles. El mostreig del contingut d'una variable, es realitzarà mitjançant l'etiqueta {{ nom-de-la-variable }}. Una mateixa variable pot contindre propietats diferents (tantes com propietats definides en la taula de la base de dades a la qual

pertanyi aquesta entitat), en aquest cas emprarem la notació {{ variable.propietat }}. Twig busca el valor de la propietat emprant les següents instruccions i ordre:

1. $variable[“propietat”]
2. $variable -> propietat
3. $variable -> propietat()
4. $variable -> getPropietat()
5. $variable -> isPropietat()
6. Null

Primerament, twig busca en la plantilla un array anomenat $variable el qual contingui una clau anomenada propietat. En cas de no trobar-la, busca un objecte anomenat $variable el qual disposi d’una propietat anomenada propietat. Si existeix l’objecte, però no la propietat, realitza una búsqueda mitjançant els *getters* més comuns (propietat(), getXXX(), isXXX()). Finalment, si no es capaç d’esbrinar el valor d’aquesta, retorna Null.

4.2.4.2 Formateig d’informació

Modificar la informació abans de mostrar-la és una acció molt extensa en la generació de plantilles twig. Aquesta acció es realitzarà mitjançant l’etiqueta {{ variable.propietat | filtre }} i els filtres predefinits:

- Striptags: elimina qualsevol etiqueta HTML que contingui la variable, a causa que aquestes poden interferir en la funcionalitat de la plantilla.
- Upper: transforma el contingut de la variable en lletres majúscules.
- Capitalize: transforma la primera lletra del text a majúscula i la resta en minúscules.
- Date (“d/m/y”, “Europe/Paris”): mostra la data en el format i regió especificat.
- Default (‘descripció alternativa’): permet assignar valor a la variable en cas que el contingut d’aquesta es trobi buit.
- Trans: permet realitzar la traducció de cadenes de text mitjançant un catàleg prèviament configurat.

4.2.4.3 Variables dinàmiques

A més de les variables les quals es renderitzen a la plantilla, twig disposa de l'etiqueta set per tal de generar-les dinàmicament, les quals són eliminades automàticament al terme de la generació d'aquesta.

- {% set variable = valor %}: genera una variable amb el nom i valor especificats.

4.2.4.4 Estructura de control if

La lògica de l'estructura de control if és similar a la de qualsevol llenguatge de programació.

- {% if usuari.connectat %}

  {# ... #}

  {% elseif %}

  {# ... #}

  {% else %}

  {# ... #}

  {% endif %}

Normalment, l'estructura if es complementa mitjanant els operadors is e is not.

- {% if reserves is divisibleby (5) %}

  {# ... #}

  {% endif %}

- {% if descripció is not empty %}

  {# ... #}

  {% endif %}

On resideix verdaderament la seva funcionalitat, és en la utilització d'operadors lògics per tal de confeccionar expressions complexes o combinar diverses expressions entre sí.

- Operadors lògics:

Operador	Explicació	Exemple
and	Retorna true solament si els dos operands de l'expressió són true	{% if usuari.registrat and usuari.edat > 18 %}
&&	Notació alternativa a l'operador and	
or	Retorna true si algun dels dos operands de l'expressió és true	{% if usuari.registrat or usuari.edat > 18 %}
||	Notació alternativa a l'operador or	

Taula 11: Operadors lògics d'estructures de control

- Operadors de comparació:

Operador	Explicació	Exemple
==	Retorna true si els dos operands són iguals	{% if usuari.edat == 18 %}
!=	Retorna true si els dos operands són diferents	{% if usuari.edat != 18 %}
>	Retorna true si el primer operand és major que el segon	{% if usuari.edat > 18 %}
<	Retorna true si el primer operand és menor que el segon	{% if usuari.edat < 18 %}
>=	Retorna true si el primer operand és major o igual que el segon	{% if usuari.edat >= 18 %}
<=	Retorna true si el primer operand és menor o igual que el segon	{% if usuari.edat <= 18 %}

Taula 12: Operadors de comparació d'estructures de control

- Operadors matemàtics:

Operador	Explicació	Exemple
+	Suma de dos números o el valor de dos variables	{{ 3 + 2 }} {{ reserves + actual }}
-	Resta de dos números o el valor de dos variables	{{ 3 + 2 }} {{ reserves - actual }}
*	Multiplicació de dos números o el valor de dos variables	{{ 3 * 2 }} {{ preu * quantitat }}
/	Divisió de dos números o el valor de dos variables	{{ 3 / 2 }}
//	Divisió entera de dos números o el valor de dos variables	{{ 3 // 2 }}
%	Mòdul de dos números o el valor de dos variables	{{ 3 % 2 }}

Taula 13: Operadors matemàtics d'estructures de control

4.2.4.4 Estructura de control for

L'ús bàsic de l'estructura for consisteix en realitzar el procés d'iteració sobre tots els elements agrupats en una col·lecció de variables. Per a l'execució d'aquesta funcionalitat, twig proveeix al programador del següent operador lògic:

- Operador contenidor: Twig defineix un operador anomenat in, el qual comprova si un valor es troba comprés per una col·lecció de variables.

```
{% for vendes in vendes %}
    {# ... #}
{% endfor %}
```

Per a que el codi presentat anteriorment funcioni correctament, no és obligatori que la variable vendes sigui un array. Simplement, és suficient que la variable sobre la que s'itera implementi la interfície Traversable o contable.

A més, twig ha dissenyat una variant anomenada for ... else, la qual implementa la mateixa lògica que l'estructura if ... else.

```
{% for vendes in vendes %}
    {# ... #}
{% else %}
    No s'ha realitzat cap compra
{% endfor %}
```

L'estructura de control for crea una variable especial anomenada loop, la qual conté informació sobre cada iteració, molt útil en la fase d'execució.

```
{% for vendes in vendes %}
    Article número: {{ loop.index }}
    Encara resten un total de {{ loop.revindex }} articles
{% endfor %}
```

Les propietats disponibles en la variable loop són les següents:

Propietat	Contingut
loop.index0	Número d'iteració, suposant que la primera sigui el número 0
loop.index	Número d'iteració, suposant que la primera sigui el número 1
loop.revindex0	Número d'iteracions que resten, suposant que la primera sigui el

	número 0
loop.revindex	Número d'iteracions que resten, suposant que la primera sigui el número 1
loop.first	Retorna true si és la primera iteració, false en cas contrari
loop.last	Retorna true si és la última iteració, false en cas contrari
loop.length	Número total d'iteracions

Taula 14: Propietats de la variable especial loop

El principal desavantatge dels bucles for de Twig respecte als de PHP, és que no disposen de mecanismes per al control d'iteracions, com break (parada del bucle) o continue (per a no realitzar una o més iteracions). Per aquest motiu és tasca del programador dur a terme la comprovació de les possibles situacions d'error que es puguin succeir.

4.2.4.5 Herència de plantilles

L'herència de plantilles segueix la mateixa dinàmica que l'herència de classes, on el seu objectiu és la reutilització de codi. En el nostre cas en particular, s'ha emprat les plantilles del frontend com a base, i la plantilla cajaLogin.html.twig com a heretadora.

```
<div id="lateral">
        {% block sidebar %}
            {% render 'UsuarioBundle:Default:cajaLogin' %}
        {% endblock %}
</div>
```

Figura 46: Herència de plantilles

En twig, les parts heretades s'anomenen blocs, i es defineixen mitjançant l'etiqueta block. Mitjançant l'objecte render, es descriu la ubicació de la plantilla a generar.

4.2.5 Control d'accés

Symfony 2, elabora una biblioteca pròpia de funcions, la qual permet l'autenticació dels usuaris que accedeixen al contingut privat de l'aplicació, mitjançant diferents tipus de xifratge o certificats X.509.

Com es va especificar en el capítol 2, l'aplicació desenvolupada es composa de tres parts (frontend públic, frontend privat (d'accés restringit a usuaris anònims) i backend (administrador)). Les pàgines que comporten el discriminador **usuario** ó **backend**, sol poden ser accedides per usuaris *loguejats*, a excepció del formulari de registre i d'accés.

4.2.5.1 Restringit l'accés

La seguretat de les aplicacions de Symfony 2 es defineix en l'arxiu de configuració app/config/security.yml. Observem la següent figura:

```
security:

    acl:
        connection: default

    firewalls:
        # Firewall de la parte de administración o backend
        backend:
            pattern:        ^/backend
            provider:       administradores
            http_basic:     ~
        # Firewall global utilizado en la parte pública o frontend
        frontend:
            pattern:        ^/*
            provider:       usuarios
            anonymous:      ~
            form_login:
                login_path: usuario_login
                check_path: usuario_login_check
            logout:
                path:       usuario_logout
            remember_me:
                key:        Paradeta1234
                lifetime:   604800  # 604.800 = 3.600 * 24 * 7 = 1 semana

    access_control:
        - { path: ^/usuario/(login|registro), roles: IS_AUTHENTICATED_ANONYMOUSLY }
        - { path: ^/usuario/*, roles: ROLE_USUARIO }
        - { path: ^/backend/*, roles: ROLE_ADMIN }

    encoders:
        Paradeta\UsuarioBundle\Entity\Usuario:        { algorithm: sha512, iterations: 10 }
        Symfony\Component\Security\Core\User\User: sha512


    providers:
        # Usuarios del frontend
        usuarios:
            entity: { class: Paradeta\UsuarioBundle\Entity\Usuario, property: email }
        # Usuarios del backend
        administradores:
            users:
                admin: { password: Eti36Ru/pWG6WfoIPiDFUBxUuyvgMA4L8+LLuGbGyqV9ATuT9brCWPch }
                                   BqX5vFTF+DgntacecW+sSGD+GZts2A==, roles: ROLE_ADMIN }

    role_hierarchy:
        ROLE_ADMIN: [ROLE_USUARIO, ROLE_ALLOWED_TO_SWITCH]
```

Figura 47: Configuració de la seguretat

A continuació es desglossa la seva configuració:

- Firewall:
    - La secció definida mitjançant la clau firewalls, es tradueix com el mecanisme mitjançant el qual es protegeix les diferents parts de l'aplicació. Cada aplicació pot definir tants firewalls com precisi, sem-

pre que se li assigni un nom únic a cadascun (en el nostre cas hem definit dos firewalls, un pel frontend i l'altre pel backend).

La funció del firewall és comprovar si la URL sol·licitada forma part de la seva definició. Més tècnicament, comprova si la URL sol·licitada coincideix amb l'expressió regular definida en l'opció pattern.

- El patró (pattern) del firewall frontend és simplement ^/, com a conseqüència totes les URL de l'aplicació es troben protegides per aquest firewall. Això comporta que qualsevol usuari el qual intenti accedir a l'aplicació s'hauria d'autenticar. Per evitar-ho, afegim l'opció anonymous: ~. En canvi, el pattern firewall backend és simplement backend/*, on les URL que emprin aquest discriminador estaran protegides pel firewall d'administrador.

  El motiu d'haver sotmès totes les URL de l'aplicació al firewall frontend, és conseqüència d'haver emprat la lògica de les versions anteriors del framework, on tota la configuració d'usuari es trobava encapsulada a l'interior d'aquest firewall.

- L'opció login_form mostra una plantilla amb un formulari d'accés per introduir les credencials.

- L'opció remember_me permet mantenir l'usuari connectat durant el transcurs d'un període de temps amb la finalitat que aquest no s'hagi d'acreditar la pròxima vegada que visiti l'aplicació.

  - L'opció Key és un valor aleatori i secret emprat per xifrar el contingut de la *cookie*, la qual s'envia a l'usuari per que aquest romangui connectat.

  - L'opció lifetime indica, en segons, el temps que l'usaria romandrà *loguejat* sense la necessitat de sol·licitar-li la contrasenya. El seu valor és 604800, degut a:

    - 3600 seg/h * 24 h /dia * 7 dia /set = 1 setmana.

- L'opció remember_me s'ha d'incloure al formulari com un camp de tipus checkbox i el seu atribut name ha de ser _remember_me.

➢ Acces_control:

- La secció definida mitjançant la clau acces_control indica quin tipus de usuari poden accedir a cada tipus d'URL. En cas del frontend, tant sols pot ser accedida per usuaris amb rol ROLE_USUARIO. L'opció acces_control és la responsable d'impedir l'accés a usuaris anònims i

el firewall és el responsable de redirigir la petició al formulari d'accés de l'aplicació.

- L'opció path és l'expressió regular la qual indica les URL a les que se l`hi aplica la configuració de seguretat.
- L'opció roles especifica el tipus d'usuaris els quals tindran accés a determinades URL de l'aplicació.

➢ Encoders:

- La secció definida mitjançant la clau encoders, especifica la codificació emprada per xifrar la contrasenya dels usuaris els quals accedeixen o es registren a l'aplicació. S'ha emprat una codificació sha512, de 10 iteracions, degut a la gran quantitat de càrrega computacional que requereix emprar la codificació estàndard, la qual es composa de 5000 iteracions.

➢ Providers:

- La secció definida mitjançant la clau providers és l'encarregada de generar els usuaris de l'aplicació. Symfony 2, defineix dos tipus de proveïdors:
  - Memory: Els usuaris es creen en memòria dinàmica amb les dades inclosos en el propi arxiu security.yml.
  - Entity: Els usuaris es creen mitjançant les entitats de Doctrine 2.

La configuració presentada en la figura 47 especifica que els usuaris es generen a partir de l'entitat Usuario del bundle UsuarioBundle i el nom d'usuari correspondrà a la propietat email de l'entitat.

4.2.5.2 Proveïdor d'usuaris

La configuració de seguretat presentada estableix que els usuaris de l'aplicació es creïn mitjançant l'entitat Usuario. No obstant, les entitats que són proveïdores d'usuaris han d'implementar la interfície UserInterface. Aquesta operació l'hem de realitzar important la seva biblioteca mitjançant la instrucció use Symfony\Component\Security\Core \User\UserInterface i els següents sis mètodes:

➢ Equals(): S'invoca quan l'aplicació necessita comprovar si un determinat usuari és igual a un altre que es referència com paràmetre del mètode. No

és necessari comprovar totes les propietats d'un usuari per determinar si son idèntics, simplement hem de comprovar aquella propietat que el fa únic, es a dir, la propietat email.

- EraseCredentials(): S'invoca quan l'aplicació requereix esborrar la informació més sensible de l'usuari, com per exemple la contrasenya.
- GetPassword(): S'invoca cada vegada que l'aplicació requereix obtenir la contrasenya de l'usuari.
- GetRoles(): S'invoca quan l'aplicació requereix obtenir un array amb tots els rols que posseeix l'usuari. S'ha prefixat el RoleUsuario com valor per defecte per aquest mètode, a causa que tots els usuaris del frontend són del mateix tipus.
- GetSalt(): Retorna el valor aleatori el qual es va emprar per xifrar la contrasenya quan es va crear l'usuari. S'invoca sempre que l'aplicació requereix comprovar la contrasenya de l'usuari.
- GetUsername(): S'invoca per obtindre el *login* que s'empra per autenticar els usuaris. D'aquesta manera, resulta molt més eficient emprar qualsevol propietat de l'entitat com un *login*, com per exemple el seu email.

4.2.5.2 Formulari de login

El procés de *login* de Symfony 2, es troba associat a tres direccions:

- /login: S'empra amb la finalitat de mostrar el formulari de *login*.
- /login_check: Acció la qual comprova que les credencials introduïdes siguin correctes.
- /logout: S'empra per desconnectar l'usuari *loguejat*.

L'única direcció la qual és obligatòria definir és /login, a causa que el formulari de *login* i l'acció que el processa s'han de crear manualment. Les direccions /login_check i /logout les processa automàticament el component de seguretat de Symfony 2, però és recomanable definir-les per tal d'aconseguir incloure-les fàcilment en les plantilles.

Per tal d'acreditar als usuaris, implementarem l'acció loginAction() en la classe src/Paradeta/UsuarioBundle/Controller/Default.php. Observem la implementació realitzada:

```
class DefaultController extends Controller
{
```

```
    public function loginAction()
    {
        $peticion = $this->getRequest();
        $sesion = $peticion->getSession();
        $error = $peticion->attributes->get(
            SecurityContext::AUTHENTICATION_ERROR,
            $sesion->get(SecurityContext::AUTHENTICATION_ERROR)
        );

        return $this->render('UsuarioBundle:Default:login.html.twig', array(
            'last_username' => $sesion->get(SecurityContext::LAST_USERNAME),
            'error'         => $error
        ));
    }
}
```

Figura 48: Acreditació d'usuaris

La clau per entendre el codi anterior, és la classe SecurityContext. Aquesta classe és el punt d'entrada al component de seguretat de Symfony 2. Mitjançant la classe s'obté el token que representa a l'usuari de l'aplicació (incloent els usuaris anònims).

La variable $error obté el valor de l'últim error produït relacionat amb la seguretat. Primerament, intenta obtenir aquesta informació mitjançant la propietat attributes de la petició. Si no la troba, busca el possible error en la sessió activa. No és obligatòria que l'acció de *login* busqui errors, però és recomanable fer-ho a causa que es millora l'experiència de l'usuari, perquè la plantilla podrà mostrar quin és exactament l'error produït (contrasenya o usuari incorrecte, ...).

Finalment, la plantilla login.html.twig és instanciada mitjançant les variables $error i $last_username. Aquesta última conté el valor de l'últim nom d'usuari emprat per intentar accedir a l'aplicació.

La plantilla necessària per mostrar el formulari de *login* és la següent:

```
{% if error %}
        <div id="ErrorLogin">{{ error.message|trans }}</div>
{% endif %}

<form action="{{ path('usuario_login_check') }}" method="post">
        <label for="username">Email</label>
        <input type="text" id="username" name="_username" value="{{ last_username }}" />

        <label for="password">Contraseña</label>
        <input type="password" id="password" name="_password" /><br><br>

        <div class="recordar">
```

```
            <input type="checkbox" id="remember_me" name="_remember_me" checked />
            No cerrar sesión
        </div>

        <div class="bott">
            <input class="boton" type="submit" name="login" value="Entrar" />
        </div>
</form>
```

Figura 49: Formulari d'accés

Les condicions que han d'acomplir tots els formularis de *login* són les següents:

- L'atribut action del formulari és la direcció la qual referència l'acció login _check. No és necessari crear aquesta acció, a causa que Symfony 2 intercepta l'enviament del formulari i s'encarrega de comprovar les credencials.
- La demarcació del nom d'usuari ha de contenir un atribut name equivalent al _username.
- La demarcació de la contrasenya ha de contenir un atribut name equivalent al _password.

4.2.5.3 Processament d'errors associats al login

Symfony2 no disposa d'una llibreria d'internacionalització de missatges d'error associats al procés de *login*, com a conseqüència, és tasca del programador definir els valors de traducció manualment.

Segons l'estàndard, la forma més ràpida i eficient de traduir els continguts d'error, consisteix en la utilització del filtre trans de twig i la creació d'un catàleg de traduccions.

4.2.5.3.1 Catàleg de traduccions

Les traduccions en Symfony2, es gestionen mitjançant catàlegs, els quals són arxius en format .xliff. Aquets arxius són els contenidors de les traduccions al idioma pertinent de les diferents cadenes de text que conformen els missatges d'error. Per definició, el nom del catàleg és messages seguit del valor *locale* (idioma de traducció predefinit en l'aplicació) i el format de l'arxiu, en el nostre cas en particular i seguint la nomenclatura estàndard, messages.es.xliff.

Aquest, es troba ubicat al directori Resources/translations/ del bundle UsuarioBundle.

```xml
<?xml version="1.0"?>
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2">
    <file source-language="en" datatype="plaintext" original="file.ext">
        <body>

        <trans-unit id="1">
                <source>The presented password cannot be empty.</source>
                <target>El campo contraseña no puede estar vacío.</target>
        </trans-unit>
        <trans-unit id="2">
                <source>Bad credentials</source>
                <target>El usuario introducido no es válido</target>
        </trans-unit>
        <trans-unit id="3">
                <source>The presented password is invalid.</source>
                <target>La contraseña introducida no es válida</target>
        </trans-unit>

        </body>
    </file>
</xliff>
```

Figura 50: Catàleg de traduccions

La seva configuració es presenta a continuació:

- L'atribut source-language='en', especifica l'idioma original dels missatges d'error.
- L'atribut trans-unit id="id", especifica la clau de búsqueda dels missatges d'error.
- L'atribut <source>, especifica el missatge d'error original.
- L'atribut <target>, especifica el missatge d'error a mostrar a l'usuari.

4.2.5.4 Obtenció de l'usuari autenticat

En un controlador, l'usuari autenticat s'obté mitjançant el servei security.context:

```php
class DefaultController extends Controller
{
    public function defaultAction()
    {
        $usuario = $this->get('security.context')->getToken()->getUser();
        $nombre = $usuario->getNombre();

        // ...
    }
}
```

Figura 51: Obtenció de l'usuari

El mètode getToken() retorna el *token* que representa a l'usuari autenticat. Emprant el mètode getUser() del *token* s'obté l'objecte de la entitat Usuari, el qual el representa.

En una plantilla, l'usuari autenticat s'obté mitjançant la propietat user de la variable global app creada per Symfony 2:

```
{% set usuario = app.user %}
Nombre: {{ usuario.nombre }}
```

Figura 52: Obtenció de l'usuari autenticat

4.2.5.5 Determinació de rols d'usuari

Com s'ha introduït en les seccions i capítols anteriors, per tal de diferenciar els tipus d'usuaris els quals tenen accés a l'aplicació, s'han definit tres rols:

- Role_Anonymously: És emprat per tots els usuaris no acreditats a l'aplicació. Aquestos sol tindran accés a la part pública del frontend i al formulari de login i registre de la part privada del frontend.
- Role_Usuario: És emprat per tots els usuaris acreditats a l'aplicació. Aquests tindran accés tant a la part pública com a la privada del frontend.
- Role_Admin: És emprat per l'administrador de l'aplicació. Aquest tindrà accés a la configuració de l'aplicació, poden manipular totes les entitats sobre la qual es conforma.

Si el contingut a mostrar depèn del tipus d'usuari, és equivalent a executar una comprovació dels determinats *roles* dels que aquest disposa. En un controlador, aquest s'obté mitjançant el mètode isGranted().

```
class DefaultController extends Controller
{
    public function defaultAction()
    {
        if ($this->get('security.context')->isGranted('ROLE_USUARIO')) {
            // el usuario tiene el role 'ROLE_USUARIO'
        }
        elseif ($this->get('security.context')->isGranted('ROLE_ADMIN')) {
            // el usuario tiene el role 'ROLE_ADMIN'
        }

        // ...
    }
}
```

Figura 53: Obtenció dels rols al controlador

El mètode isGranted() és molt fàcil d'emprar, però la seva lògica de funcionament és força complexa i genera una alta càrrega computacional, motiu pel qual hem de mini-

mitzar el seu ús el màxima possible. Si l'usuari no disposa de l'autorització necessari per accedir a la URL sol·licitada, podem mostrar-li una pàgina d'error mitjançant l'excepció AccessDeniedException.

En una plantilla twig, el *role* d'usuari s'obté mitjançant el mètode is_granted(), el qual es comporta d'igual forma que el mètode isGranted() dels controladors.

```
{% if is_granted('ROLE_USUARIO') %}
    {# ... #}
{% elseif is_granted('ROLE_ADMIN') %}
    {# ... #}
{% endif %}
```

Figura 54: Obtenció dels rols en una plantilla

L'únic inconvenient de la funció is_granted() és la utilització d'aquesta en una plantilla la qual la seva URL no es trobi protegida per cap tipus de *firewall*, cosa que comporta que es produeixi un error i twig retorni una excepció.

4.2.5.6 Codificació de la contrasenya d'usuari

La configuració de seguretat presentada, xifra la contrasenya d'usuari mitjançant l'algoritme sha512. Aquesta operació es realitza emprant la configuració definida a l'arxiu security.yml.

```
class DefaultController extends Controller
{
    public function defaultAction()
    {
        $usuario = new Usuario();
        $encoder = $this->get('security.encoder_factory')
                        ->getEncoder($usuario);

        $password = $encoder->encodePassword(
            'la-contraseña-en-claro',
            $usuario->getSalt()
        );

        $usuario->setPassword($password);
    }
}
```

Figura 55: Algoritme de codificació de la contrasenya

L'objecte $encoder, obtingut mitjançant el servei security.encoder_factory, és l'encarregat de codificar la contrasenya. Com arguments, es referència l'objecte de l'usuari per al qual es requereix la realització de l'operació.

El mètode encodePassword() xifra la contrasenya mitjançant la variable Salt, la qual es un valor aleatori emprat per xifrar la contrasenya mitjançant l'algoritme presentat anteriorment.

4.2.6 Interacció amb la base de dades

La manipulació de la informació de Doctrine2 (buscar, crear, modificar i eliminar registres de la taula de la base de dades), es realitza mitjançant l'objecte especial anomenat entity manager. L'estàndard de definició d'aquest és el següent:

- $em = $this->getDoctrine()->getEntityManager(): Funció emprada per a la vinculació de la consulta amb l'entity manager.
- $producto = $em->getRepository('ProductoBundle:Producto')->metode_de_busqueda(): Funció associada a l'acció de consulta. La variable $producto conté totes les propietats de l'entitat associada a la búsqueda.

Els mètodes que aquest empra per a la búsqueda d'informació són els següents:

Mètode	Funció
find(1)	Realitza una búsqueda en la taula associada a l'entitat pel valor definit en el filtre.
findAll()	Realitza la búsqueda de totes les entitats pertanyents a un mateix tipus, és a dir, totes les files de la taula.
findBy(array(filtre_búsqueda))	Realitza la búsqueda de les propietats de l'entitat que compleixin el filtre associat en l'array de búsqueda.
findOneBy (array(filtre_búsqueda))	Realitza un única búsqueda de la propietat de l'entitat que compleixi el filtre associat en l'array de búsqueda.

Taula 15: Propietats de búsqueda

4.2.6.1 Mètodes de búsqueda personalitzats

Doctrine2, al igual que Symfony2, és força restrictiu i els seus mètodes de búsqueda no permeten la realització d'accions complexes, com per exemple un join. Per aquest motiu, es realitzarà un repositori propi de búsqueda. L'estàndard defineix l'entitat com a vinculació entre els mètodes de búsqueda pertanyents a l'aplicació i els personalitzats.

```
/**
 * Paradeta\UsuarioBundle\Entity\Usuario
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="Paradeta\UsuarioBundle\Entity\UsuarioRepository")
 */
```

Figura 56: Definició d'un repositori de búsqueda

A la capçalera de definicions d'ORM de Doctrine2 de l'entitat Usuari, es defineix la classe UsuarioRepository com un mètode personalitzat de búsqueda.

```php
<?php

namespace Paradeta\UsuarioBundle\Entity;

use Doctrine\ORM\EntityRepository;

class UsuarioRepository extends EntityRepository
{
    /**
     * Encuentra todas las compras del usuario indicado
     *
     * @param string $usuario indica el id del usuario
     */
    public function findTodasLasCompras($usuario)
    {
        $em = $this->getEntityManager();

        $fecha = new \DateTime('now - 8 week');

        $consulta = $em->createQuery('SELECT v, p FROM ProductoBundle:Venta v
JOIN v.producto p WHERE v.usuario = :id AND v.fecha_recogida >= :fecha ORDER BY
v.fecha_recogida DESC, v.tipo DESC');
        $consulta->setParameter('id', $usuario);
        $consulta->setParameter('fecha', $fecha);

        return $consulta->getResult();
    }

}
```

Figura 57: Mètode de búsqueda personalitzat

La funció findTodasLasCompras() realitza una búsqueda per usuari i data recent de compra. La seva configuració és la següent:

- Es parametritza l'identificador de l'usuari autenticat en l'aplicació mitjançant la variable $usuario.
- Mitjançant el mètode createQuery(), es crea la consulta emprant el llenguatge DQL. Per evita el *lazy loading*, s'inclou l'identificador de l'usuari en la consulta.
- El mètode setParameter() estableix el valor del filtres de búsqueda (id de l'usuari i antiguitat màxima de recollida).
- L'antiguitat màxima de recollida és de 8 setmanes anteriors al dia vigent, degut a que no es vol col·lapsar a l'usuari amb informació antiga.
- El mètode getResult() retorna els valors de la consulta realitzada, la qual ja la trobem disponible al controlador.

4.2.6.2 Creació, modificació i eliminació d'informació

Les operacions d'inserció, modificació i eliminació, també es troben associades a l'*entity manager*.

4.2.6.2.1 Creació d'informació

El següent exemple mostra com crear una entitat de tipus Venda, per a una acció posterior d'emmagatzematge.

```
$venta = new Venta();
$venta->setFechaRecogida(new \DateTime('now'));
$venta->setProducto($producto);
$venta->setUsuario($usuario);

// Guardar la nueva compra en la base de datos
$em->persist($venta);
$em->flush();
```

Figura 58: Processament d'entitat Venda

Processar la creació i l'emmagatzematge, és tan simple com instanciar la classe corresponent a l'entitat Venda, on mitjançant els *setters* es definirà la informació corresponent a aquesta, i posteriorment gracies a l'*entity manager* s'invocarà el mètode persist(), parametritzant-li l'entitat sobre la qual es vol realitzar l'operació d'inserció.

El mètode persist() marca l'entitat com *persistente* (creada i definida en memòria), però no l'emmagatzema en la base de dades (realitzant un símil amb el llenguatge SQL, no executa l'operació insert). On radica la finalitat d'aquesta acció és en la dràstica reducció de la càrrega computacional que genera la constant execució de sentencies *DQL*, degut a que Doctrine2 no executa cap sentencia fins que es sol·licita explícitament mitjançant el mètode flush(), és a dir, no realitza una inserció per a cada propietat de l'entitat, sinó que l'emmagatzema en memòria dinàmica fins que el controlador sol·licita l'acció d'emmagatzematge.

4.2.6.2.2 Modificació d'informació

Modificar les dades d'una entitat, és molt similar a la creació. La diferencia radica en que no s'opera mitjançant una entitat buida, sinó que es processa a partir d'una existent obtinguda mitjançant el pertinent mètode de búsqueda.

```
$usuario = $this->get('security.context')->getToken()->getUser();
// Si el usuario no ha cambiado el password, su valor es null después
```

```
// de hacer el ->bindRequest(), por lo que hay que recuperar el valor original
if (null == $usuario->getPassword()) {
    $usuario->setPassword($passwordOriginal);
}
// Si el usuario ha cambiado su password, hay que codificarlo antes de guardarlo
else {
    $encoder = $this->get('security.encoder_factory')->getEncoder($usuario);
    $passwordCodificado = $encoder->encodePassword(
        $usuario->getPassword(),
        $usuario->getSalt()
    );
    $usuario->setPassword($passwordCodificado);
}

$em->persist($usuario);
$em->flush();
```

Figura 59: Modificació de l'entitat Usuari

Com podem observar en la figura 59 , primerament es busca la informació referent a l'entitat Usuari, i acte seguit, es realitza el conjunt d'operacions pertinents.

4.2.6.2.3 Eliminació d'informació

L'operació d'eliminació de les dades referents a una entitat es realitza mitjançant el mètode remove(). El següent exemple il·lustra com buscar un usuari especificat per a una posterior eliminació d'aquest.

```
$em = $this->getDoctrine()->getEntityManager();

$usuario = $em->getRepository('UsuarioBundle:Usuario')->findByDni('1234567L');

$em->remove($usuario);
$em->flush();
```

Figura 60: Eliminació de l'entitat Usuari

Tècnicament, el funcionament de Doctrine2 descrit anteriorment i l'obligació d'emprar el mètode flush() per emmagatzemar els canvis, es coneix com patró *Unit of Work*. Aquest patró respon a l'acció de "*mantindre una llista d'objectes modificats en el transcurs d'una transacció i coordinar la persistència d'aquestos canvis i la resolució dels possibles problemes associats a la concurrencia*".

4.2.7 Elaboració de formularis

Els formularis són els encarregats d'intervenir la interacció entre usuari – aplicació. Aquestos es creen mitjançant l'objecte *form builder*, obtingut a partir del mètode

createFormBuilder(), el qual parametritza un objecte amb les dades inicials les quals es mostraran al formulari.

```
public function registroAction()
{
    $usuario = new Usuario();

    $formulario = $this->createFormBuilder($usuario)
        ->add('nombre')
        ->add('apellidos')
        ->add('direccion', 'text')
        ->add('fechaNacimiento', 'date')
    ->getForm();

    return $this->render(
        'UsuarioBundle:Default:registro.html.twig',
        array('formulario' => $formulario->createView())
    );
```

Figura 61: Elaboració del formulari d'Usuari

Observant la figura 61, un cop obtingut el *form builder*, s'afegeix els camps a mostrar del formulari mitjançant el mètode add(), el qual disposa dels següents tres arguments:

- El primer argument és el **nom** del camp, el qual ha de ser únic, a causa que aquest és emprat com a identificador al formulari.
- El segon argumente opcional e indica el **tipus** de camp emprat per a representar-lo. Per defecte, sinó s'indica el contrari, tots els camps són de tipus text i es representen com <input type="text" />.
- El tercer argument també és opcional i consisteix en un array amb les **opcions** emprades per crear el camp.

La classe *form builder* és de tipus *"fluent interface"*, la qual permet encadenar totes les crides als seus mètodes. Tots els mètodes de la classe inclouen la següent instrucció al final, return $this;, la qual renderitza la plantilla a mostrar a l'usuari.

Els mètode add(), simplement afegeixen els camps al formulari. Per a crear pròpiament l'objecte que representa el formulari i les seves dades inicials, s'invoca el mètode getform(). Aquest objecte no es pot referenciar directament a la plantilla, com a conseqüència, s'ha d'invocar el mètode createView(), el qual elabora la representació visual de cada camp amb la informació disponible i les opcions establertes.

4.2.7.1 Creació d'un repositori propi

Crear el formulari a l'interior del controlador és la forma més ràpida i simple de generar-lo. L'inconvenient és que aquest no es pot reutilitzar en cap altra part de l'aplicació.

L'estàndard de Symfony2 recomana definir una classe per cada formulari. Aquestes classes es troben definides per defecte al directori Form/ del *bundle*, el nom del qual finalitza amb la terminació Type (en el nostre cas en particular, s'ha emprat el *bundle* UsuarioBundle com a base de repositori). Seguint la nomenclatura, es crearan tres classes diferents de formularis:

- UsuarioType.php, emprat per registrar usuaris.
- ReservaType.php, emprat per gestionar reserves.
- VentaType.php, emprat per gestionar vendes.

A continuació es descriu únicament la configuració de la classe UsuarioType.php, a causa que totes les elaborades segueixen la mateixa nomenclatura i estructura.

```php
<?php

namespace Paradeta\UsuarioBundle\Form\Frontend;

use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilder;
use Doctrine\ORM\EntityRepository;

class UsuarioType extends AbstractType
{
    public function buildForm(FormBuilder $builder, array $options)
    {
        $builder
            ->add('nombre')
            ->add('apellidos')
            ->add('email', 'email')
            ->add('password', 'repeated', array(
                'type' => 'password',
                'label' => 'Autenticación',
                'first_name' => 'Contraseña',
                'second_name' => 'Repite la contraseña',
                'invalid_message' => 'Las dos contraseñas deben coincidir',
                'required' => true
            ))

            ->add('telefono')
            ->add('direccion')
            ->add('fecha_nacimiento', 'birthday')
            ->add('dni')

        ;
    }
```

```
    public function getName()
    {
        return 'Paradeta_usuariobundle_usuariotype';

    }
}
```

Figura 62: Configuració de la classe UsuarioType

La classe presentada hereta d'AbstractType, la qual actua de formulari base. La funció getName() retorna el nom únic del formulari. La descripció de la seva nomenclatura és deguda a {nom del projecte}_{nom del bundle}_{nom de la classe}.

El formulari s'elabora mitjançant el mètode buildForm(), al qual Symfony2 referència com a primer argument un *form builder* i com a segon un array amb les opcions de creació. Emprant el mètode add() s'afegeixen totes les propietats a manipular de l'entitat.

El controlador processa el nou repositori mitjançant el mètode createForm(), disponible a tots els controladors els quals hereten de la classe Controller.

```
public function registroAction()
{
    $usuario = new Usuario();
    $usuario->setFechaNacimiento(new \DateTime('today - 18 years'));
    $formulario = $this->createForm(new UsuarioType(), $usuario);

    // ...
    return $this->render(
        'UsuarioBundle:Default:registro.html.twig',
        array('formulario' => $formulario->createView())
    );
}
```

Figura 63: Parametrització de la classe UsuarioType

Com primer argument, es referència la classe del formulari. Opcionalment, com segon argument, es referència l'objecte amb les dades inicials a mostrar a l'usuari. La finalitat d'aquest és definir la data de naixement, per defecte, 18 anys anteriors al dia vigent d'avui; en el cas del formulari de compra la data de recollida, per defecte, serà vigent a dia d'avui.

4.2.7.2 Formateig dels camps

Per a l'elaboració de l'aplicació, s'ha emprat les següents especificacions de camps:

- Camp email, aquest es mostra a l'usuari com un camp de correu electrònic. Visualment, mostra l'aparença d'un quadre de text, però la seva lògica pre-

configurada comprova que el correu compti amb el format definit segons l'estàndard.

- Camp repeated, Symfony2 disposa d'aquest mètode especial el qual genera dos camps iguals (dos camps de contrasenya) i comprova la igualtat del valor introduït. El tercer argument del mètode add() especifica les opcions de creació del camp:
  - Type, indica la tipologia dels dos camps a crear (password, cosa que comportarà l'ocultació dels valors que introdueixi l'usuari).
  - Invalid_message, estableix el missatge d'error que es mostrarà en cas que els dos valors introduïts per l'usuari no coincideixin.
  - First_name / Second_name, especifica el nom del camp a mostrar en cada cas.
  - Required, especifica que és un valor obligatori.
- Camp birthday, restringeix el rang de valors que poden esser seleccionats per l'usuari. Aquest rang estipula 120 anys anteriors al dia vigent.
- Camp choice, genera una llista desplegable, on els valors disponibles es troben indicats en una col·lecció prèviament configurada sobre un array, el qual es parametritza com a tercer valor del camp.

```
->add('tipo','choice', array(
      'choices'  => array ('d' => 'Comer', 'c' => 'Cenar')))
```

Figura 64: Especificació del camp tipus choice

4.2.7.2 Formateig de la informació

El formulari s'insereix en la plantilla mitjançant el mètode formAction(). La configuració d'aquest és la següent:

- Path, indica el nom de la direcció lògica del formulari.
- Method, indica la tipologia d'accions del formulari, és a dir, post (accions produïdes a causa de la interacció amb l'usuari).
- Form_enctype, indica el nom únic descrit al controlador, és a dir, formulario

```
<form action="{{ path('usuario_registro') }}" method="post" {{ form_enctype(formulario) }}>
        <div>
                {{ form_errors(formulario) }}

                <div>
```

```
            {{ form_row(formulario.nombre) }}
        </div>

        <div>
            {{ form_label(formulario.apellidos) }}
            {{ form_errors(formulario.apellidos) }}
            {{ form_widget(formulario.apellidos) }}
        </div>

        <div>
            {{ form_errors(formulario.password) }}
            {{ form_widget(formulario.password) }}

            <p class="ayuda">Escribe dos veces la misma contraseña.</p>
        </div>

        // . . .

        <div>
            {{ form_label(formulario.dni, 'DNI') }}
            {{ form_errors(formulario.dni) }}
            {{ form_widget(formulario.dni, { 'attr': { 'class': 'corto' } }) }}

            <p class="ayuda">Escribe las 8 cifras y la letra sin dejar ningún espacio en blanco</p>
        </div>

        {{ form_rest(formulario) }}

        <p class="ayuda">Al pulsar en "Registrarme", aceptas los términos especificados en el apartado <a
>Aviso Legal</a> y las <a href="{{ path('condiciones') }}">Condiciones</a> de uso</p>
        <div id="botondes">
            <input class="boton" type="submit" value="Registrarme" />
        </div>

    </div>
```

Figura 65: Plantilla del formulari de registre

Els camps presentats a continuació de la descripció del formulari, es troben destinats al mostreig d'errors i missatges d'ajuda, modificació de títols, ... A continuació es desglossa la seva configuració:

- Form_label (formulari.propietat), mostra el títol del camp (etiqueta <label>).
- Form_errors (formulari.propietat), mostra els errors específics de cada camp.
- Form_widget (formulari.propietat), mostra la etiqueta HTML necessària per a la representació del camp.

4.2.7.3 Processant el formulari

Symfony2 empra la següent lògica de processament, un únic controlador s'encarrega de mostrar el formulari inicial i de processar l'enviament de dades. El controlador empra els següents dos objectes per determinar l'acció a realitzar:

- GET, creació d'un formulari vuit i renderitzat mitjançant la plantilla.
- POST, obtenció de les dades enviades per l'usuari, validació d'aquestes i emmagatzematge en la base de dades. Aquesta acció també compren la re-

direcció a pagines d'error ó confirmació.

Symfony2, proveeix al programador de mètodes automàtics de processament d'objectes GET i POST.

```
$peticion = $this->getRequest();
$em = $this->getDoctrine()->getEntityManager();

$usuario = new Usuario();
$usuario->setFechaNacimiento(new \DateTime('now - 18 years'));

$formulario = $this->createForm(new UsuarioType(), $usuario);

if ($peticion->getMethod() == 'POST') {

    $formulario->bindRequest($peticion);

    if ($formulario->isValid()) {

        // Processament de la lògica d'emmagatzematge

    }
}

return $this->render('UsuarioBundle:Default:registro.html.twig', array(
    'formulario' => $formulario->createView()
));
```

Figura 66: Processament del formulari

- El mètode bindRequest() associa l'objecte $formulari amb totes les dades enviades per l'usuari a la $peticion. L'objecte associat al formulari ($usuario), conté les dades remeses per l'usuari.
- El mètode isValid(), comprova si les dades del formulari acompleixen les regles de validació. En cas afirmatiu, retorna true i processa la lògica definida al seu interior. En cas contrari, retorna flase amb la consegüent acció per part de Symfony2, el qual generarà un nou formulari amb les dades remeses per l'usuari i els missatges d'error produïts.
- També és possible modificar la informació remesa per l'usuari en l'acció de POST mitjançant els *setters*, sempre i quan no s'hagi completat l'acció de flush().

4.2.8 Processant de la lògica de control

La lògica de control, compren totes les accions i restriccions definides en els requeriments funcionals. Aquestos són de tres tipus:

- **Temporals**, comprovaran si la data de recollida o la de reserva és adient.
- **Espacials**, comprovaran si la gestió web no supera el màxim previst d'afora-

ment possible.

- **Alimentaris**, comprovarà la impossibilitat de l'acció de compra de peix i marisc amb data de recollida diferent al cap de setmana.

4.2.8.1 Restriccions temporals

La interacció amb les compres i les reserves realitzades pels usuaris, es troben sotmeses a regles de validació temporals pròpies, on el seu rang de valors compren des del dia vigent fins a cinc anys posteriors a aquest. Si el dia d'interacció seleccionat és igual al dia vigent, les restriccions a aplicar són les següents:

- Les compres i reserves amb franja horària d'interacció per dinar, l'hora màxima de formalització d'aquestes són les 13:00.

```
$comida = \DateTime::createFromFormat ('H',13);
$comida1 = $comida -> getTimestamp();
```

Figura 67: Formateig del límit de dinar

- Les compres i reserves amb franja horària d'interacció per sopar, l'hora màxima de formalització d'aquestes són les 20:00.

```
$cena = \DateTime::createFromFormat ('H',20);
$cena1 = $cena -> getTimestamp();
```

Figura 68: Formateig del límit de sopar

Les dos figures presentades anteriorment realitzen l'operació de formateig de l'hora màxima especificada en el disseny funcional. Mitjançant el mètode createFromFormat() es realitza la creació de l'hora màxima parametritzant com primer paràmetre l'objecte sobre el qual realitzar l'operació (H) i com a segon paràmetre el valor d'aquest. El mètode \DateTime crea la data actual amb els valors especificats al constructor. El mètode getTimestamp() realitza la conversió de la cadena de string, format actual de la data generada, a una cadena d'int per tal de realitzar la posterior comparació temporal.

El motiu d'haver generat una cadena d'string, és conseqüència que no és possible realitzar una comparació lògica (no es pot determinar si un string és major o menor, temporalment, a un altre; tant sols es pot realitzar l'operació d'igualtat).

Conseqüentment, emprant el constructor temporal de Symfony2, es defineix un enter amb la data i hora actual.

```
$ahora = new \DateTime();
```

```
$ahora1 = $ahora -> getTimestamp();
```

Figura 69: Formateig de l'hora actual

Definits els comparadors temporals, es realitzarà el processament lògic per tal de determinar si els valors introduïts en la compra o la reserva són vàlids.

```
if ($formulario->isValid()) {

        // Si la fecha de recogida es vigente a día de hoy
        if ($reserva -> getFechaReserva() == new \DateTime('today')) {

                // Si la franja de recogida es la de comer
                if ($reserva -> getTipo() == 'd') {

                        // Si la hora es mayor a las 13:00
                        if ($ahora1 > $comida1) {

                                // Lanzamos un mensaje de error, con su correspondiente ayuda
                                return $this->redirect($this->generateUrl('error1'));
                        }
                }

                // Si la franja de recogida es la de cenar
                elseif ($reserva -> getTipo() == 'c') {

                        // Si la hora es mayor a las 20:00
                        if ($ahora1 > $cena1) {

                                // Lanzamos un mensaje de error, con su correspondiente ayuda
                                return $this->redirect($this->generateUrl('error2'));
                        }
                }
        }
```

Figura 70: Processament de la lògica temporal

Si el formulari és vàlid, es determinarà si la data introduïda en aquest és igual al dia vigent. En cas afirmatiu, es realitzarà la comparació temporal; en cas contrari, es processarà l'emmagatzematge d'aquest en la base de dades. La restricció temporal comprovarà si la franja horària (camp tipo) seleccionada és igual a dinar (d) ó sopar (c), amb la conseqüent comparació horària. Si el valor introduït supera el límit horari, es redireccionarà a l'usuari al missatge d'error pertinent.

4.2.8.2 Restriccions espacials

La interacció amb les reserves realitzades pels usuaris es troben sotmeses a regles de validació pròpies, on l'aforament màxima especificat per gestions online és de 50 comensals.

La lògica que aquest ha de comprendre es detalla a continuació:

```
$suma = $em->getRepository('ProductoBundle:Reserva')->findTotalReservas($fecha, $tipo);
$total = 0;
foreach ($suma as $suma) {
```

```
        $suma1 = $suma -> getComensales();
        $total = $suma1 + $total;
}
$auxiliar = $reserva -> getComensales();
$total = $total + $auxiliar;
if ($total > 50) {
        return $this->redirect($this->generateUrl('error3'));
}
```

Figura 71: Processament de la lògica espacial

Per a l'elaboració de la lògica de control pertinent, es realitzarà una búsqueda sobre la qual es parametritzarà la data de reserva i la franja horària a la qual pertany aquesta (la variable $suma contindrà un array amb el resultat d'aquesta). Mitjançant un bucle es recorrerà l'array retornat, on la variable $total contindrà el total de comensals de la búsqueda realitzada. Finalment, emprant la variable $auxiliar, s'afegirà al total emmagatzemat en variable $total, els comensals especificats en la reserva realitzada per l'usuari. En cas afirmatiu, es processarà l'emmagatzematge d'aquesta en la base de dades; en cas contrari, es retornarà a l'usuari el missatge d'error pertinent.

4.2.8.3 Restriccions alimentaries

La interacció amb les compres realitzades pels usuaris es troben sotmeses a regles de validació pròpies, on el sortit de marisc i peixos tant sols pot ser adquirit els caps de setmana. És realitzarà la mateixa dinàmica que les temporals, però amb la següent variant.

```
if ($producto -> getClase() == 'P'){
        $fecha = $venta -> getFechaRecogida();
        $fecha1 = $fecha -> getTimestamp();
        $fecha2 = date( 'd-m-Y', $fecha1);
        $fecha3 = date('Y-m-d', strtotime($fecha2));
        $dia=date("w", strtotime($fecha3));
        if ($dia == 1 || $dia == 2 || $dia == 3 || $dia == 4 || $dia == 5) {
                return $this->redirect($this->generateUrl('usuario_error5'));
        }
}
```

Figura 72: Processament de la lògica alimentaria

La variable $fecha1, contindrà l'enter el qual representa la data introduïda per l'usuari. Aquesta es formatarà en l'estàndard (dia / mes / any) mitjançant el mètode date() de PHP. Pot semblar que s'ha realitzat dos operacions inútils al convertir la data a enter i després un altre cop d'enter a data, però el motiu d'aquesta acció és conseqüència que PHP no reconeix com a tipus data els valors retornats per Symfony2. La funció strtotime de PHP formatarà la data de l'estàndard anglès (any / mes / dia) com a data UNIX, on finalment la variable $dia contindrà el dia de la setmana (W) al qual pertany la data introduïda. En cas que el dia de la setmana sigui igual al valors 6 ó 7 (dissabte o

diumenge respectivament), es processarà l'emmagatzematge d'aquesta en la base de dades; en cas contrari, es retornarà a l'usuari el missatge d'error pertinent.

4.3 Disseny del Backend

En aquesta secció es presentarà informació sobre l'elaboració de la zona d'adminitració de l'aplicació.

4.3.1 Definició de l'encaminament

S'ha emprat la següent metodologia d'encaminament, seguint la lògica de casos d'us.

- Direcció associada a la portada:
  - backend_:
    - pattern: /backend
    - defaults: { _controller: BackendBundle:Default:backend}
- Direccions associades a la gestió de productes:
  - backend_listar_productos:
    - pattern: /backend/listar_productos
    - defaults: { _controller: BackendBundle:Default:listar_productos}
  - backend_datos_producto:
    - pattern: /backend/datos_producto/{alimento}
    - defaults: {_controller: BackendBundle:Default:datos_producto}
  - backend_buscar_producto:
    - pattern: /backend/buscar_producto
    - defaults: {_controller:BackendBundle:Default:buscar_producto}
  - backend_producto_nuevo:
    - pattern: /backend/producto_nuevo

        - defaults: {_controller:BackendBundle:Default:productoNuevo}

    - backend_producto_modificar:
        - pattern: /backend/producto_modificar/{alimento}
        - defaults: {_controller:BackendBundle:Default:productoModificar}

    - backend_producto_eliminar:
        - pattern: /backend/producto_eliminar/{alimento}
        - defaults :{_controller:BackendBundle:Default:productoEliminar}

- Direccions associades a la gestió d'usuaris:
    - backend_listar_usuarios:
        - pattern: /backend/listar_usuarios
        - defaults: {_controller:BackendBundle:Default:listar_usuarios}
    - backend_buscar_usuario:
        - pattern: /backend/buscar_usuario
        - defaults:{_controller:BackendBundle:Default:buscar_usuario}
    - backend_datos_usuario:
        - pattern: /backend/datos_usuario/{user}
        - defaults: {_controller:BackendBundle:Default:datos_usuario}
    - backend_detalles_usuario:
        - pattern: /backend/detalle_usuario/{user}
        - defaults: {_controller:BackendBundle:Default:detalle_usuario}

- Direccions associades a la gestió de vendes:
    - backend_listar_ventas:
        - pattern: /backend/listar_ventas
        - defaults: {_controller:BackendBundle:Default:listar_ventas}

- backend_detalles_venta:
  - pattern: /backend/detalle_venta/{date}
  - defaults: {_controller:BackendBundle:Default:detalle_venta}
- backend_buscar_ventas:
  - pattern: /backend/buscar_ventas
  - defaults: {_controller:BackendBundle:Default:buscar_ventas}

➢ Direccions associades a la gestió de reserves:

- backend_listar_reservas:
  - pattern: /backend/listar_reservas
  - defaults: {_controller:BackendBundle:Default:listar_reservas}
- backend_buscar_reservas:
  - pattern: /backend/buscar_reservas
  - defaults: {_controller:BackendBundle:Default:buscar_reservas}
- backend_detalles_reserva:
  - pattern: /backend/detalle_reserva/{date}
  - defaults: {_controller:BackendBundle:Default:detalle_reserva}
- backend_datos_reserva:
  - pattern: /backend/datos_reserva/{reservation}
  - defaults: {_controller: BackendBundle:Default:datos_reserva}

➢ Direccions associades als missatges de confirmació:

- backend_confirmacion8:
  - pattern: /backend/confirmacion8
  - defaults: {_controller:BackendBundle:Default:confirmacion8}

    Article emmagatzemant satisfactòriament.
- backend_confirmacion9:

- pattern: /backend/confirmacion9
- defaults: {_controller:BackendBundle:Default:confirmacion9}

  Article modificat satisfactòriament.

- backend_confirmacion10:
  - pattern: /backend/confirmacion10
  - defaults: {_controller:BackendBundle:Default:confirmacion 10}

    Article eliminat satisfactòriament.

➢ Direccions associades als missatges d'error:

- backend_error8:
  - pattern: /backend/error8
  - defaults: {_controller:BackendBundle:Default:error8}

    Error com a conseqüència d'haver seleccionat l'id d'un article inexistent en la base de dades.

- backend_error9:
  - pattern: /backend/error9
  - defaults: {_controller:BackendBundle:Default:error9}

    Error produït al seleccionar l'id d'un article inexistent en la base de dades.

- backend_error10:
  - pattern: /backend/error10
  - defaults: { _controller: BackendBundle:Default:error10}

  Error com a conseqüència d'haver seleccionat l'id d'una reserva inexistent en la base de dades.

- backend_error11:
  - pattern: /backend/error11
  - defaults: {_controller:BackendBundle:Default:error11}

Error com a conseqüència d'haver seleccionat l'id d'un usuari inexistent en la base de dades.

- backend_error12:
  - pattern: /backend/error12
  - defaults: {_controller:BackendBundle:Default:error12}

    Error produït al introduir el nom d'un article inexistent en la base de dades en el procés de búsqueda.

4.3.2 Control d'accés

El firewall definit en la secció 4.2.5.1, determina el punt d'accés a l'àrea d'administrador com http_basic. Aquesta opció mostra la *caixa login* del propi navegador, la qual sol·licitarà les credencials a l'administrador.

4.3.3 Creació d'un repositori propi

El catàleg d'articles es complementarà amb la creació d'un repositori propi, el qual permeti l'edició d'aquestos. Es crearà una nova classe anomenada ProductoType.php.

```
class ProductoType extends AbstractType
{
    public function buildForm(FormBuilder $builder, array $options)
    {
        $builder
            ->add('nombre')
            ->add('descripcion')
            ->add('foto', 'file', array('required' => false))
            ->add('precio', 'money')
            ->add('clase','choice', array(
                  'choices'  => array ('S' => 'Sidra', 'V' => 'Vino', 'A' => 'Cava', 'T' => 'Tapa',
                  'P' => 'Pescado', 'C' => 'Carne', 'E' => 'Entrante')));

    }

    public function getName()
    {
        return 'Paradeta_backendbundle_productotype';

    }
}
```

Figura 73: Configuració de la classe ProductoType

4.3.3.1 Formateig dels camps

S'ha emprat les següents especificacions de camps:

- Camp file, aquest es mostra a l'administrador com "*entrada d'arxius*". Aquest camp instanciarà la classe UploadedFile, la qual comprovarà la correcta introducció d'una imatge.
- Camp money, aquest es mostra a l'administrador com moneda, el qual comprovarà que el valor introduït sigui de tipus enter amb una precisió de dos decimals.
- Camp choice, genera una llista desplegable, on els valors disponibles es troben indicats en una col·lecció d'elements representatius d'articles.

4.3.4 Processament d'imatges

Les imatges del catàleg d'articles podran ser modificades lliurement. Aquesta operació es realitzarà mitjançant el mètode subirFoto() definit en l'entitat Producte.

```
public function subirFoto($directorioDestino)
{
    if (null === $this->foto) {
        return;
    }

    $nombreArchivoFoto = $this->foto->getClientOriginalName();

    $this->foto->move($directorioDestino, $nombreArchivoFoto);

    $this->setFoto($nombreArchivoFoto);
}
```

Figura 74: Definició del mètode subirFoto()

Aquest mètode hereta de la classe UploadedFile, la qual genera el camp del formulari de tipus file. El mètode getClientOriginalName() clona el nom de la imatge a la variable $nombreArchivoFoto. El mètode move copia la imatge física al directori destí amb el nom definit anteriorment en la variable. La variable $directorioDestino es parametritza prèviament al controlador. Finalment s'emmagatzemarà en la propietat Foto el valor definit en la variable $nombreArchivoFoto.

Les imatges del catàleg d'articles s'emmagatzemaran en el directori uploads/images/, ubicat al directori públic de l'aplicació (web). L'estàndard recomana encabir aquesta informació en l'arxiu de configuració propi de l'aplicació (config.yml) i posteriorment realitzar una crida d'aquest en el controlador.

```
parameters:
    paradeta.directorio.imagenes: %kernel.root_dir%/../web/uploads/images/
```

Figura 75: Definició de l'arxiu config.yml

La lògia definida al controlador és la següent:

```
if ($producto -> getFoto() != NULL) {
        $producto->subirFoto($this->container->getParameter('paradeta.directorio.imagenes'));
}
if ($producto -> getFoto() == NULL) {
        $producto ->setFoto('no_disponible.jpg');
}
```

Figura 76: Processament d'imatges

Si l'administrador introdueix una imatge al formulari de creació d'articles, es realitzarà la crida al mètode subirFoto(), parametritzant el directori d'imatges mitjançant el mètode $this->container->getParameter('paradeta.directorio.imagenes'). En cas contrari, s'emprarà la imatge per defecte.

4.3.4 Parametrització d'imatges

La parametrització d'imatges en un plantilla pot ser una tasca sovint força complexa, degut a l'emmagatzematge de tant sols el nom d'aquesta. Per a l'obtenció de la ruta relativa d'aquesta, s'ha emprat la funció asset() de twig ,mitjançant la qual es retorna la ruta pública (relativa al directori /web/ del projecte) de l'arxiu indicat.

```
<img alt="Fotografía de la oferta" src="{{ asset(directorio imagenes ~ producto.foto) }}">
```

Figura 77: Parametrització d'imatges

La funció asset(), parametritza com primer element el directori on es troben ubicades les imatges de l'aplicació, i com a segon el nombre d'aquesta.

5 Proves Unitàries

Les proves unitàries o proves de components són un tipus de proves de programari, la finalitat de les quals és realitzar un anàlisis dels components o unitats exhaustives de codi font per tal de verificar la correcta execució de l'aplicació i poder detectar els possibles *bugs* generats en l'entorn d'execució, on finalment enviar una aplicació lliure d'errors al servidor de producció.

5.1 Anàlisis de funcionalitats

Idealment, cada prova ha de ser independent de les altres, a causa que si el programador no contempla una situació d'error, aquesta propagarà per l'estructuració de l'aplicació, essent una àrdua tasca la resolució d'aquesta.

Aquestes s'han estructurat seguint la lògica d'actors definits en el capítol 4, secció 4.1.1.1.

5.1.1 Proves d'usuari anònim

Les proves d'usuari anònim contemplen la correcta visualització de continguts públics, la traça idònia de la ruta especificada per l'usuari, realització de reserves, a més de realitzar el registre i accés a l'aplicació.

- Visualització de la portada:
  - Acció: L'usuari introduirà la ruta corresponent a la portada.
  - Resposta: L'aplicació processarà i mostrarà la plantilla corresponent a la portada.

Figura 78: Visualització de la portada

- Visualització de les tapes:
  - Acció: L'usuari introduirà la ruta corresponent a la secció de tapes.
  - Especificació: Degut a que el catàleg d'articles és similar en totes les pàgines elaborades, el document tant sols contindrà la prova d'una classe pertanyent a aquesta.
  - Resposta: L'aplicació processarà i mostrarà la plantilla corresponent a aquesta.

Figura 79: Plana web corresponent a les tapes

- Visualització de la localització:
  - Acció: L'usuari introduirà la ruta corresponent a la localització, e indicarà una ruta a traçar.
  - Resposta 1: L'aplicació processarà i mostrarà la plantilla corresponent a aquesta, on l'usuari introduirà una ruta errònia.

Figura 80: Definició d'una ruta errònia

  - Resposta 2: La ruta introduïda és vàlida, conseqüentment l'aplicació traçarà la ruta sobre el mapa.

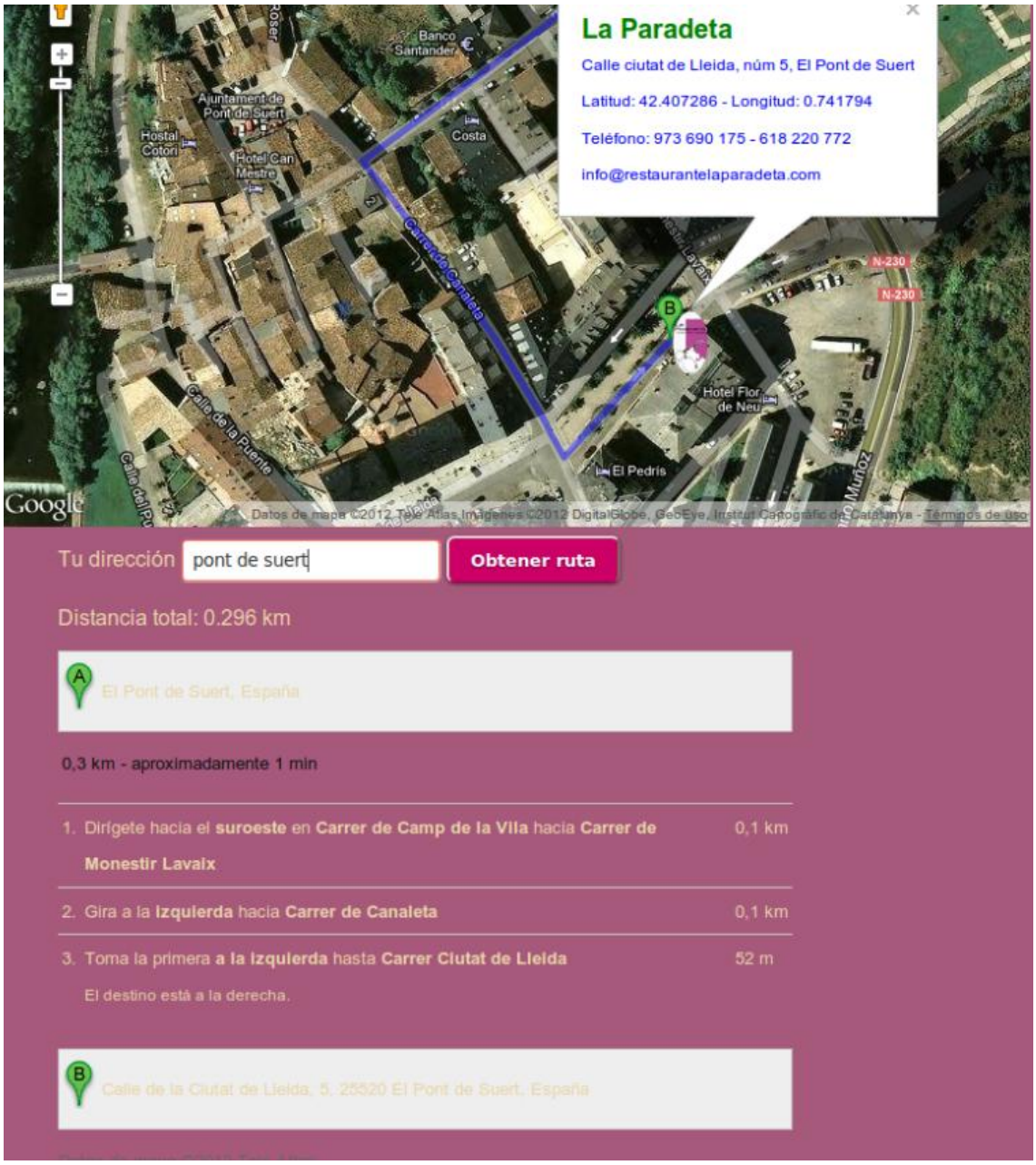


Figura 81: Traça de la ruta especificada

- Visualització del formulari de contacte:
  - Acció: L'usuari introduirà la ruta corresponent al formulari de contacte e interaccionarà amb el formulari.
  - Especificació: Degut que l'aplicació encara es troba al servidor de d'execució (es treballa amb la localhost), es processarà l'enviament d'emails mitjançant una compta de correu electrònic prèviament configurada.
  - Resposta 1: L'aplicació processarà i mostrarà la plantilla corresponent a aquesta.

Figura 82: Interacció amb el formulari de contacte

- Resposta 2: Es comprovarà el correcte enviament de les dades i el mostreig d'un missatge de confirmació a l'usuari.

Figura 83: Processament correcte de l'enviament del formulari

- Resposta 3: Es verificarà la rebuda de l'email en la compta de correu personal del programador.

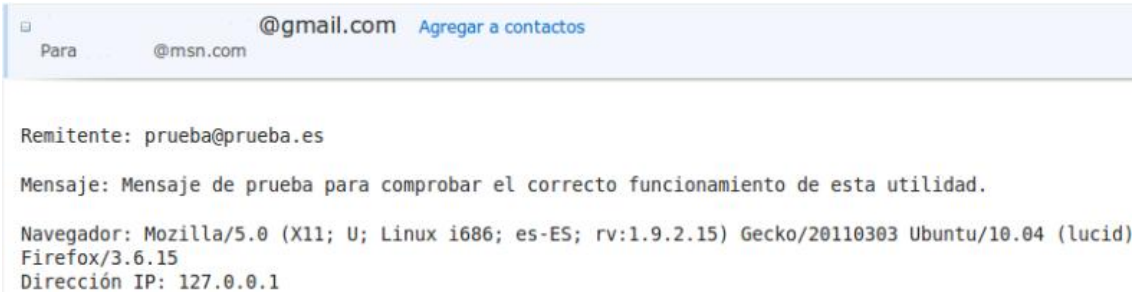


Figura 84: Rebuda del missatge enviat mitjançant l'aplicació web

- Realització de reserves:
  - Acció: L'usuari introduirà la ruta corresponent a les reserves, on es comprovarà el correcte processament de les dades d'aquesta.
  - Especificació: Degut a que la lògia de comprovació és idèntica al registre d'usuaris, el document tant sols contindrà la comprovació del formulari de reserva.
  - Resposta 1: L'aplicació comprovarà la correcta introducció de les dades.

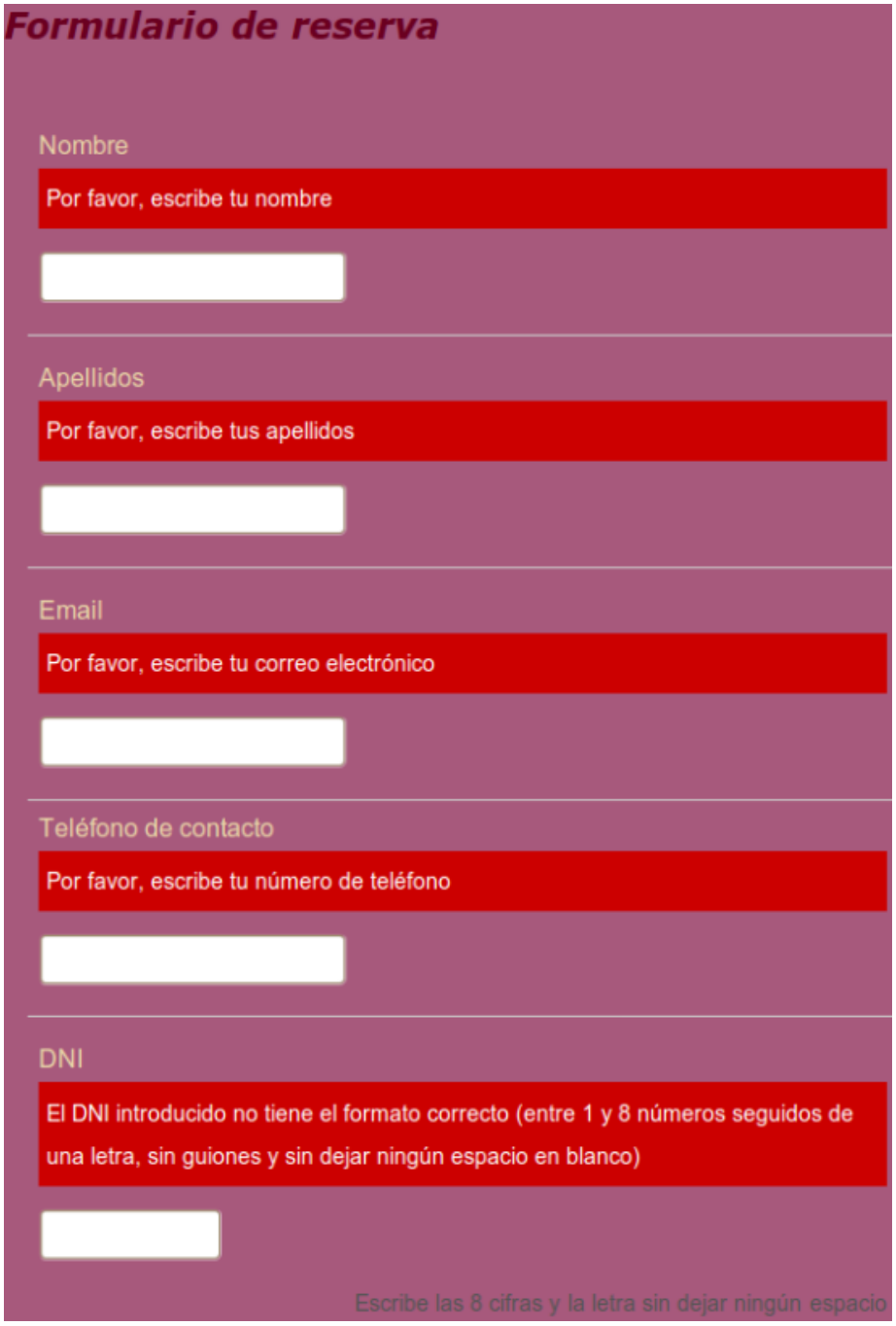
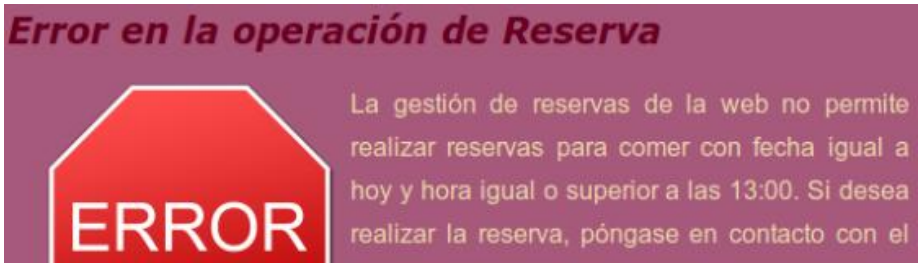


Figura 85: Formulari amb dades errònies

- Resposta 2: L'aplicació processarà el formulari, la qual mostrarà un error al haver superat l'hora màxima de reserves per dinar.

Figura 86: Reserva amb hora errònia per dinar



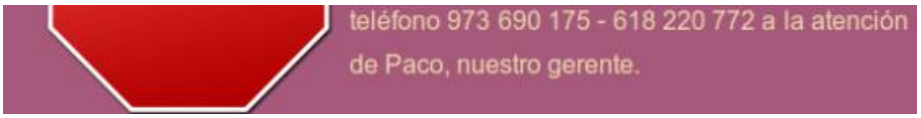


Figura 87: Processament erroni de reserva per dinar

- Resposta 3: L'aplicació processarà el formulari, la qual mostrarà un error al haver superat l'hora màxima de reserves per sopar.

Figura 88: Reserva amb hora errònia per sopar

Figura 89: Processament erroni de reserva per sopar

- Resposta 4: És superarà l'aforament màxim per al dia i franja horària especificat per l'usuari. És mostrarà un missatge d'error.

id	nombre	apellidos	email	telefono	dni	fecha_alta	fecha_reserva	tipo	comensales	comentario
12	Jorge	Gomez Arriba	jorge@alumnes.udl	649	457	2012-08-26 11:13:31	2012-08-26	d	1	Ninguna
14	Jorge	Gomez Arriba	jorge@alumnes.udl	649	457	2012-08-26 11:43:21	2012-08-26	d	49	Ninguna

Figura 90: Contingut de la taula Reserva per dia i horari especificats

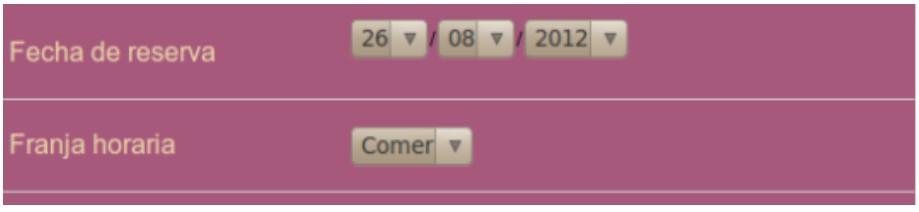


Figura 91: Reserva errònia, la qual supera el límit d'aforament

Figura 92: Processament erroni de la reserva degut al límit espacial

- Resposta 6: L'aplicació processarà el formulari, on els valors introduïts seran adients. És mostrarà un missatge de confirmació.

  - Reserves per dinar.

Figura 93: Processament satisfactori de reserves per dinar

  - Reserves per sopar.

Figura 94: Processament satisfactori de reserves per sopar

- Formulari d'accés:
  - Acció: L'usuari introduirà la ruta corresponent al registre, on es comprovarà les credencials d'aquest, introduint valors erronis.
  - Resposta 1: L'aplicació comprovarà la correcta introducció de les dades, on l'usuari introduït no correspon a cap registrat a l'aplicació.

Figura 95: Usuari no vàlid

  - Resposta 2: L'aplicació comprovarà la correcta introducció de les dades, on l'usuari és correcte, però la contrasenya és incorrecta.

Figura 96: Contrasenya no vàlida

5.1.2 Proves d'usuari acreditat

La finalitat d'aquesta és comprovar la correcta nomenclatura en la definició de l'encaminament, la visualització i modificació del perfil, la realització i visualització de compres i visualització de reserves.

- Realització de compres:
  - Acció: L'usuari introduirà la ruta corresponent a aquesta e interaccionarà amb l'aplicació.

- Especificació: Les restriccions temporals han estat comprovades al formulari de reserva, conseqüentment únicament es realitzarà la comprovació de les restriccions alimentaries.
- Resposta: L'aplicació processarà el formulari, la qual mostrarà un error al haver seleccionat un article corresponent a la secció de marisc amb una data de recollida diferent de cap de setmana.

Figura 97: Data de recollida de marisc errònia

Figura 98: Processament erroni de la reserva degut a la restricció de alimentaria

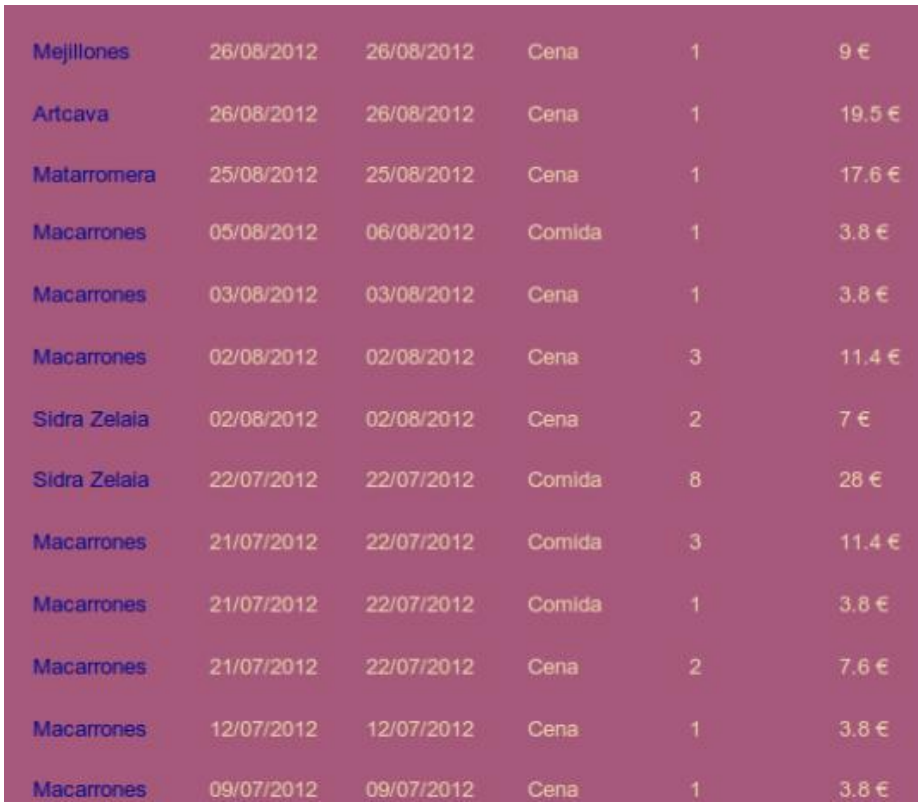
- Visualització de compres:
  - Acció: L'usuari introduirà la ruta corresponent a la visualització.
  - Resposta: L'aplicació processarà la petició, la qual mostrarà una taula amb les compres més recents (superiors a 2 mesos anteriors al dia vigent).
  - Especificació: S'emmarcaran en color vermell les compres que es mostraran a l'usuari.

id	producto_id	usuario_id	fecha_compra	fecha_recogida	tipo	cantidad
1	1	1	2012-06-21 17:29:38	2012-06-21	c	4
2	1	1	2012-06-21 17:29:45	2012-06-21	c	3
3	1	1	2012-06-21 17:32:18	2012-06-21	c	3
4	1	1	2012-06-23 18:33:14	2012-06-23	c	2
5	1	1	2012-06-23 18:34:55	2012-06-23	c	1
6	1	1	2012-06-23 18:38:55	2012-06-23	c	1
7	1	1	2012-06-28 18:57:58	2012-07-02	c	4
8	1	1	2012-06-28 19:03:27	2012-06-29	d	5
9	1	1	2012-07-09 17:06:25	2012-07-09	c	1
10	1	1	2012-07-12 18:14:14	2012-07-12	c	1
11	1	1	2012-07-21 13:53:17	2012-07-22	d	1
12	1	1	2012-07-21 13:53:28	2012-07-22	c	2
13	1	1	2012-07-21 13:53:34	2012-07-22	d	3
14	2	1	2012-07-22 11:58:37	2012-07-22	d	8
16	2	1	2012-08-02 17:16:50	2012-08-02	c	2
17	1	1	2012-08-02 17:23:06	2012-08-02	c	3
18	1	1	2012-08-03 16:39:16	2012-08-03	c	1
19	1	1	2012-08-05 17:39:11	2012-08-06	d	1
20	4	1	2012-08-25 19:10:33	2012-08-25	c	1
21	7	1	2012-08-26 11:19:19	2012-08-26	d	1
22	7	1	2012-08-26 11:19:47	2012-08-26	c	1
23	10	1	2012-08-26 11:21:22	2012-08-26	d	1
24	10	1	2012-08-26 11:21:42	2012-08-26	c	1
25	2	1	2012-08-26 13:30:29	2012-08-26	c	1
26	1	1	2012-08-26 13:34:42	2012-08-26	c	1
27	8	1	2012-08-26 13:35:20	2012-08-26	c	3

Figura 99: Contingut de la taula Venta

Tus compras

Nombre del artículo	Fecha de compra	Fecha de recogida	Horario de recogida	Total de artículos	Importe total
Mejillones	26/08/2012	26/08/2012	Comida	1	9 €
Artcava	26/08/2012	26/08/2012	Comida	1	19.5 €
Choricillos a la sidra	26/08/2012	26/08/2012	Cena	3	13.5 €
Macarrones	26/08/2012	26/08/2012	Cena	1	3.8 €
Sidra Zelaia	26/08/2012	26/08/2012	Cena	1	3.5 €



Mejillones	26/08/2012	26/08/2012	Cena	1	9 €
Artcava	26/08/2012	26/08/2012	Cena	1	19.5 €
Matarromera	25/08/2012	25/08/2012	Cena	1	17.6 €
Macarrones	05/08/2012	06/08/2012	Comida	1	3.8 €
Macarrones	03/08/2012	03/08/2012	Cena	1	3.8 €
Macarrones	02/08/2012	02/08/2012	Cena	3	11.4 €
Sidra Zelaia	02/08/2012	02/08/2012	Cena	2	7 €
Sidra Zelaia	22/07/2012	22/07/2012	Comida	8	28 €
Macarrones	21/07/2012	22/07/2012	Comida	3	11.4 €
Macarrones	21/07/2012	22/07/2012	Comida	1	3.8 €
Macarrones	21/07/2012	22/07/2012	Cena	2	7.6 €
Macarrones	12/07/2012	12/07/2012	Cena	1	3.8 €
Macarrones	09/07/2012	09/07/2012	Cena	1	3.8 €

Figura 100: Visualització de les compres de l'usuari

5.1.3 Proves d'administrador

- La finalitat d'aquesta és comprovar la correcta nomenclatura en la definició de l'encaminament i la gestió de tota la funcionalitat presentada.
- Visualització d'articles:
  - Acció: L'administrador introduirà la ruta corresponent a aquest.
  - Resposta 1: L'aplicació processarà i mostrarà la plantilla corresponent a aquesta.

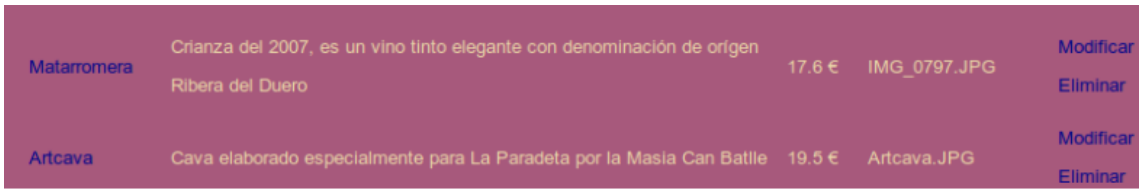


Figura 101: Portada de la gestió d'articles

- Resposta 2: L'administrador sol·licitarà un article inexistent en la base de dades, amb el consegüent missatge d'error.

Figura 102: Búsqueda d'un article inexistent

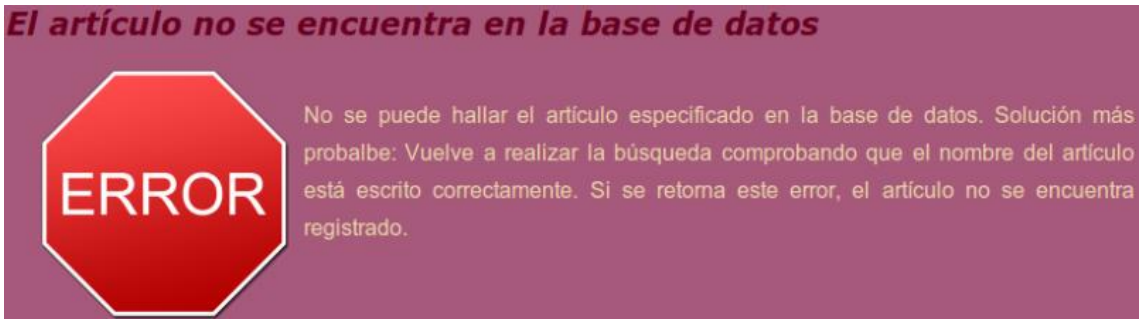
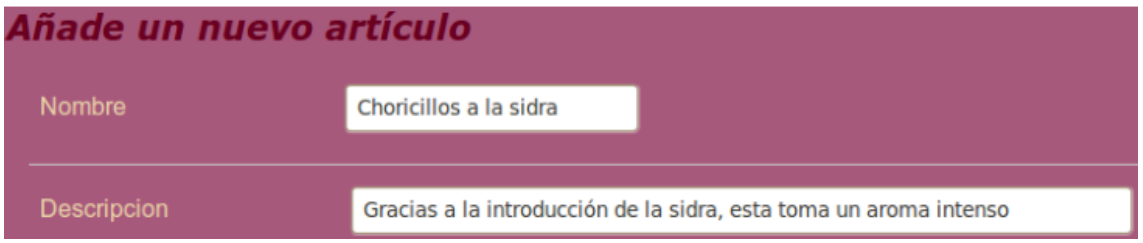


Figura 103: Processament erroni de la búsqueda de l'article

- Creació d'articles:
  - Acció: L'administrador introduirà la ruta corresponent a aquest e interactuarà amb l'aplicació.
  - Especificació: Degut a que la lògia de modificació i eliminació és idèntica a la de creació, el document tant sols contindrà la creació d'articles.
  - Resposta 1: L'aplicació processarà i mostrarà la plantilla corresponent a aquesta, on l'administrador introduirà les noves dades.



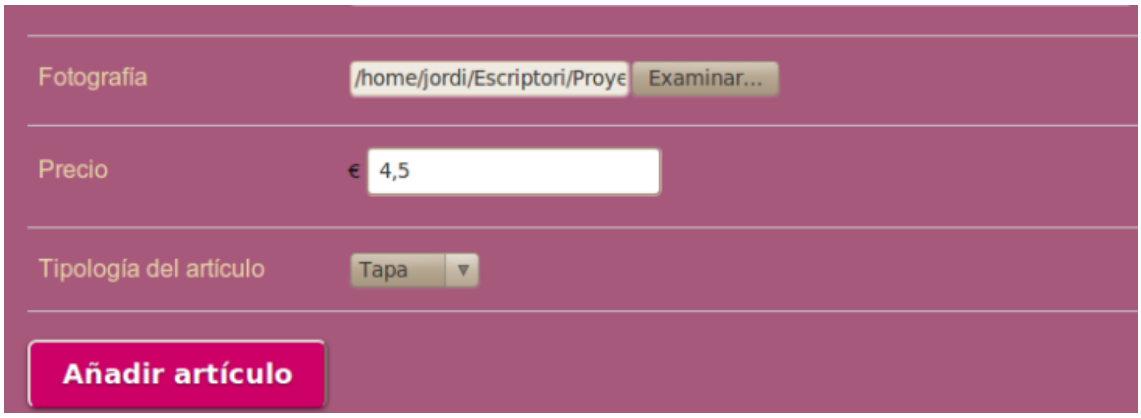


Figura 104: Creació d'un nou article

- Resposta 2: Es crearà l'article nou, on és verificarà la correcta visualització d'aquest.

Figura 105: Fitxa de l'article "choricillos a la sidra"

- Visualització de compres:
  - Acció: L'administrador introduirà la ruta corresponent a aquest.
  - Resposta 1: L'aplicació processarà i mostrarà la plantilla corresponent a aquesta.

Artcava	jorge@alumnes.udl	Jorge	649809049	26/08/2012	Cena	1
Mejillones	jorge@alumnes.udl	Jorge	649809049	26/08/2012	Comida	1
Mejillones	jorge@alumnes.udl	Jorge	649809049	26/08/2012	Cena	1

Figura 106: Portada de la gestió de compres

5.1.4 Compatibilitat de navegació

La compatibilitat de navegació defineix la visualització de l'aplicació desenvolupada en els diferents navegadors els quals es troben a disposició de l'usuari. Per realitzar aquesta tasca, s'emprarà el mashup browershot, disponible en la direcció http://browsershots.org/. A continuació es detalla la visualització als navegadors més populars.

- Mozilla firefox: Versió actual, 15.0.

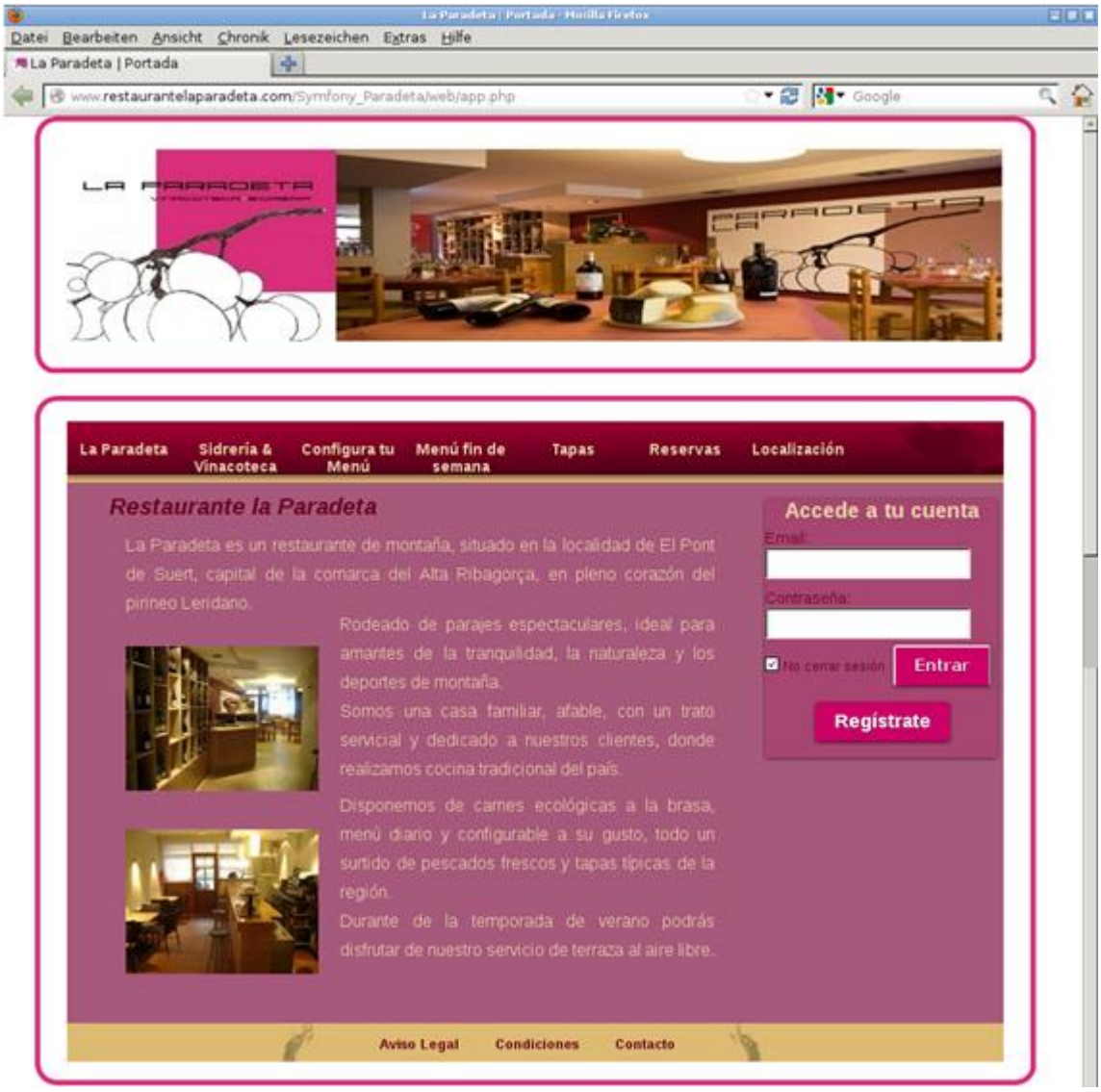


Figura 107: Visualització de l'aplicació en el navegador Firefox

- Google Chrome: Versió actual, 21.0.1180.83

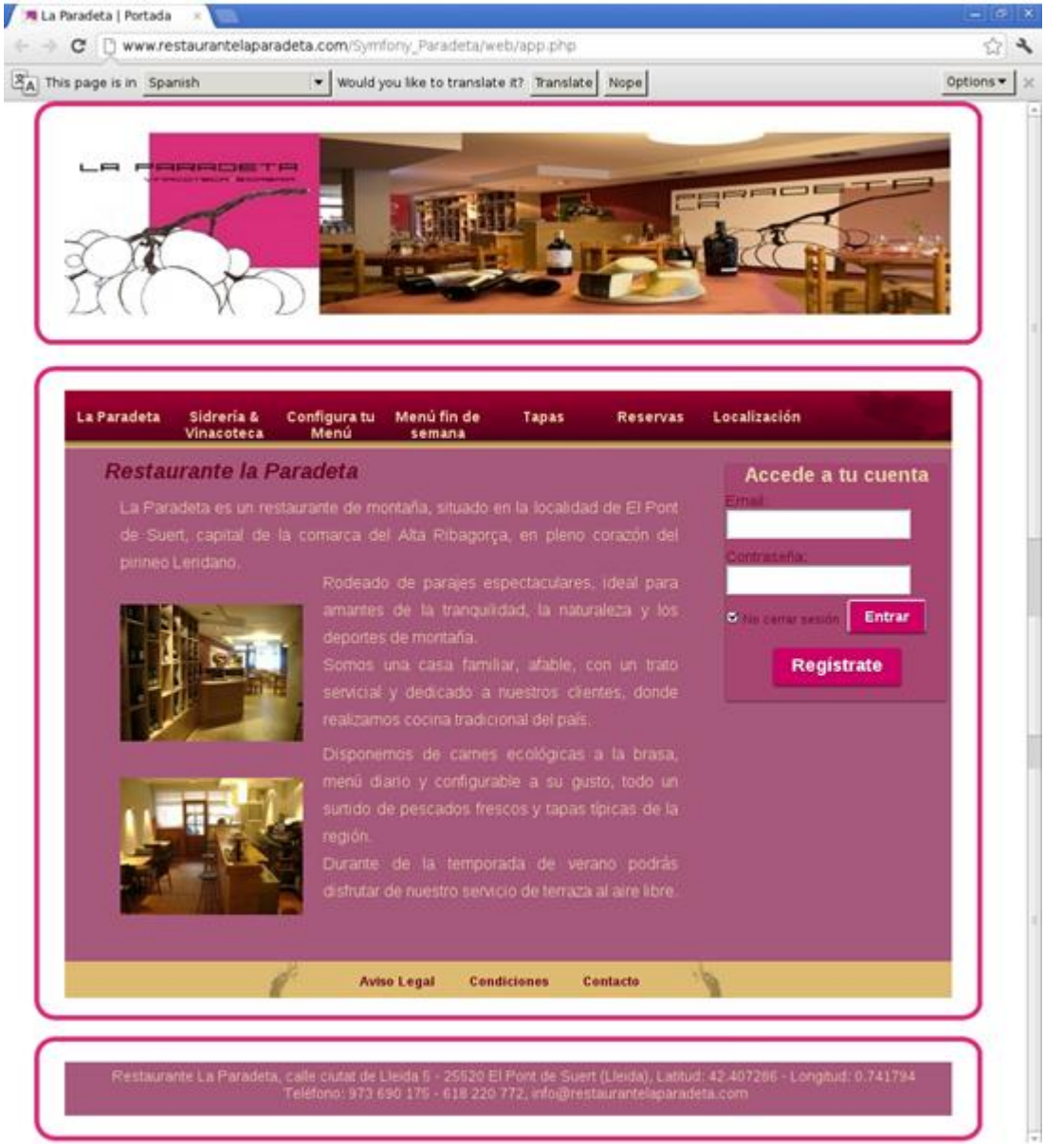


Figura 108: Visualització de l'aplicació en el navegador Chrome

- Opera: Versió actual, 12.01

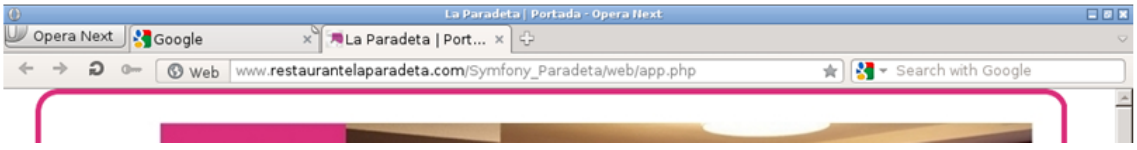


Figura 109: Visualització de l'aplicació en el navegador Opera

- Internet Explorer: Versió actual, 9.0.

Figura 110: Visualització de l'aplicació en el navegador Explorer

A causa que el mashup presentat anteriorment no suporta cap versió del navegador de Microsoft Internet explorer, s'ha emprat el mashup netrenderer, disponible en la direcció http://netrenderer.com/index.php.

- Safari: Versió actual, 5.1.

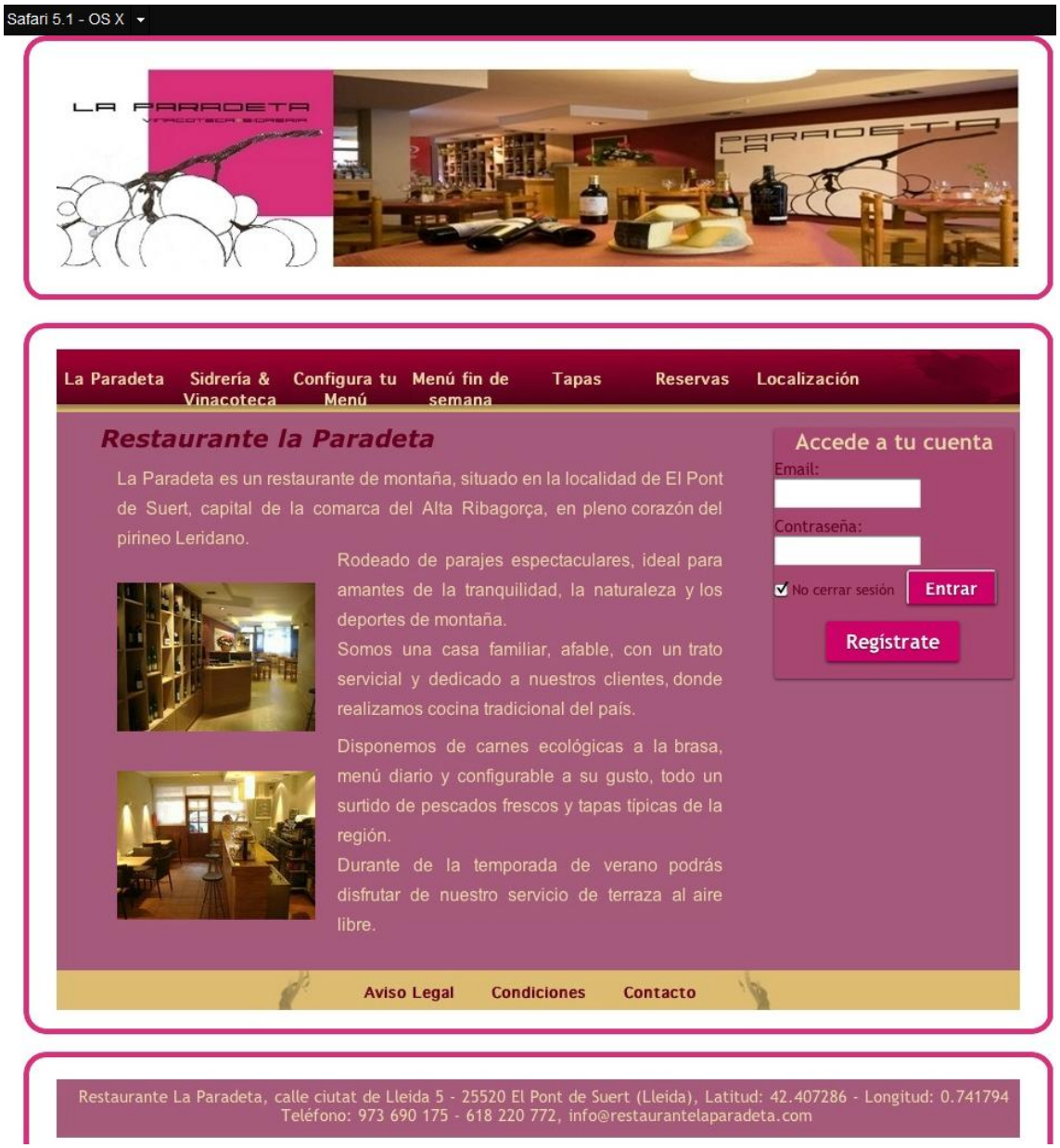


Figura 111: Visualització de l'aplicació en el navegador Safari

A causa que el conjunt de mashups presentats anteriorment no disposen de suport per al navegador de Mac Safari, s'ha emprat el mashup adobe browserlab, disponible en la direcció https://browserlab.adobe.com/es-es/index.html.

6. Conclusions i treballs futurs

En el present capítol es realitzarà una valoració a nivell personal, on es remarcaran les conclusions extretes, i s'especificarà el possibles treballs futurs i tasques de manteniment de l'aplicació.

6.1 Conclusions

Al terme del present projecte final de carrera, podem afirmar que s'ha acomplert el 100% de les especificacions esmentades en l'anàlisi de funcionalitats.

En un primer anàlisi, es va estimar una durada del desenvolupament d'aquest entorn als 5 mesos, on al final s'ha denotat com una estimació molt optimista. La durada final d'aquest ha estat d'aproximadament 7 mesos, degut al lent procés d'auto aprenentatge de les diferents eines emprades durant l'elaboració d'aquest.

Durant l'elaboració d'aquest s'ha realitzat un estudi del framework i eines associades, tals com:

- Estudi e implementació d'eines de disseny gràfic.
- Estudi e implementació de base de dades MySQL.
- Estudi e implementació del llenguatge PHP.
- Estudi e implementació del servei Twig.
- Estudi e implementació dels servis de Doctrine2.
- Estudi e implementació del llenguatge HTML5.
- Estudi e implementació d'un estil propi mitjançant la fulla d'estils CSS.
- Masterització del framework Symfony2.
- Estudi e implementació de l'API v3 de Google Maps.

A nivell personal, l'objectiu principal era introduir-me en la programació i metodologia de disseny web.

A priori la utilització d'un framework pot restar mèrit al programador a l'hora de presentar la funcionalitat final de l'aplicació, però durant el transcurs del procés d'aprenentatge ha estat un handicap, a causa que m'he hagut d'adaptar a la metodologia i lògica de funcionament d'aquest, on sovint ha estat una tasca força abstracta ja que l'estructuració d'aquest és radicalment oposada als llenguatges de programació apre-

sos durant la titulació. Tot i això, ha estat una experiència força enriquidora, amb un resultat completament satisfactori, el qual m'ha il·lustrat una nova visió de la programació orientada a objectes.

Symfony2 és un framework força transparent al programador el qual permet la realització d'un desenvolupament àgil i eficient d'aplicacions, on la funcionalitat principal és dur al programador a un entorn de treball que el permeti centrar-se en la definició de la lògica de negoci i no haver de preocupar-se per la implementació de tasques força repetitives.

La definició del firewall i les restriccions funcionals han estat les parts més costoses d'implementar, a causa de la manca d'exemples pràctics disponibles a Internet. El manual de Symfony2 sovint manca d'aquestos, on els exemples presentats composen petites avaluacions de la teoria referenciada en cada secció.

Finalment m'agradaria afirmar que l'ús d'un framework, tal com Symfony2, està altament recomanat en la realització d'aplicacions de força envergadura. Personalment, el *feedback* ha estat molt positiu, on programadors d'arreu de la xarxa sempre es troben disponibles per a la realització i resolució de consultes.

6.2 Treballs Futurs

La tecnologia avança a passos agegantats, i el disseny web no n'és una excepció. En un futur pròxim, es preveu la introducció de nous continguts disponibles en l'aplicació.

La internacionalització de l'aplicació als diferents idiomes es una tasca imminent, a causa que el model de negoci presentat pel gerent és l'atracció del major nombre de públic possible.

S'implementarà un mòdul de pagament mitjançant targeta de crèdit. Aquesta apartat no s'ha pogut dur a terme durant el transcurs del projecte a causa de la mancança de medis i coneixements per al desenvolupament d'aquest.

A nivell personal, s'han adquirit els coneixements i confiança necessària per dur a terme tot tipus d'aplicacions amb serveis orientats al disseny web.

Annex A: Instal·lació de Symfony2

La novetat més significativa de Symfony2 respecte a versions anteriors és el codi d'aquest, el qual és publicat mitjançant distribucions. En aquest aspecte, segueix la mateixa implementació que Linux. Actualment, trobem tres distribucion:

- La distribució **estandard** (http://github.com/symfony/symfony-standard), inclou un projecte ja creat, un instal·lador/configurador via web i un arxiu de configuració simple en format *.ini.
- La distribució **hello world** (http://github.com/symfony/symfony-hello-world), la qual inclou el codi font i un petit exemple que mostra la cadena de text *hello world*.
- La distribució **repositorio** (http://github.com/symfony/symfony), la qual tant sol inclou el codi font de Symfony2. Aquesta, sol es recomana per a programadors experts.

Per programadors inexperts es recomana emprar la distribució estàndard, a causa el mètode d'instal·lació és interactiu i visual, a més, mostrar el procés d'instal·lació amb tota la informació possible.

Descarregant Symfony2

La descarrega de la distribució amb **vendors** inclou tant el codi font de Symfony2 com el total de les llibreries externes necessàries (Twig, Swiftmailer, Doctrine2, ...).

A la fi de la descarrega de symfony2, hem d'assegurar-nos que l'equip emprat per al desenvolupament de l'aplicació disposa dels recursos necessaris. Per a la realització d'aquesta comprovació, entrarem al directori arrel del projecte i executarem el següent script:

```
$ php app/check.php
```

Figura 112: Script de comprovació de requisits de software

L'script check.php mostra per consola una llista de requisits obligatoris (*Mandatory requirements*) i una de requisits desitjables (*Optional checks*). En cas de fallida d'aquesta,

caldrà solucionar l'error, a causa que són requisits indispensables per al correcte funcionament de Symfony2.

Acte seguit d'aquesta comprovació, executarem el següent script amb la finalitat de verificar la correcta instal·lació de Symfony2.

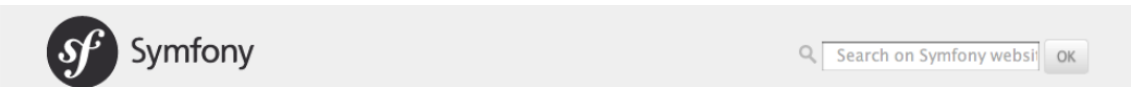
```
$ php app/console
Symfony version 2.0.x - app/dev/debug

Usage:
  [options] command [arguments]

Options:
  --help           -h Display this help message.
  --quiet          -q Do not output any message.
  --verbose        -v Increase verbosity of messages.
  --version        -V Display this program version.
  --ansi           -a Force ANSI output.
  --no-interaction -n Do not ask any interactive question.
  --shell          -s Launch the shell.
  --env            -e The Environment name.
  --debug          -d Whether to run in debug mode.

Available commands:
  help                          Displays help for a command (?)
  list                          Lists commands
assetic
  :dump                         Dumps all assets to the filesystem
assets
  :install
```

Figura 113: Script de verificació de correcta instal·lació de Symfony2

Resulta imprescindible que l'script php app/console s'executi correctament, a causa que Symfony2 empra aquestes sentencies per automatitzar la gran majoria de les tasques associades a la lògica de negoci.

Configuració de l'entorn d'execució

La URL d'entrada a l'aplicació és http://localhost/Symfony_Paradeta/web/app_dev.php, on si hem acomplert correctament el procés de configuració, hauríem de visualitzar la següent pàgina:



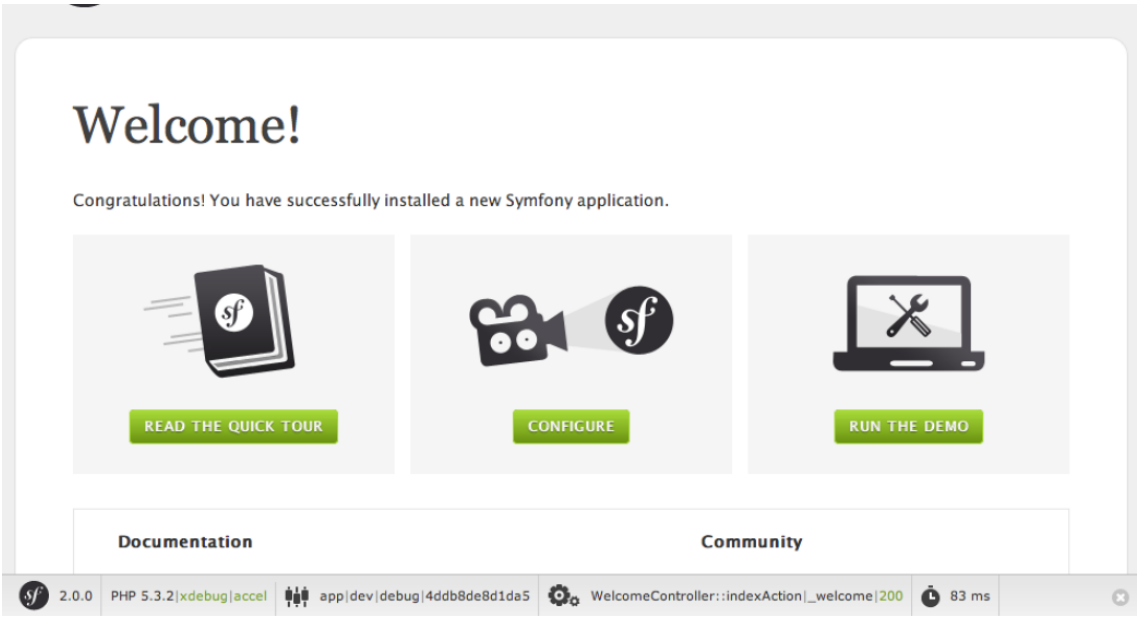
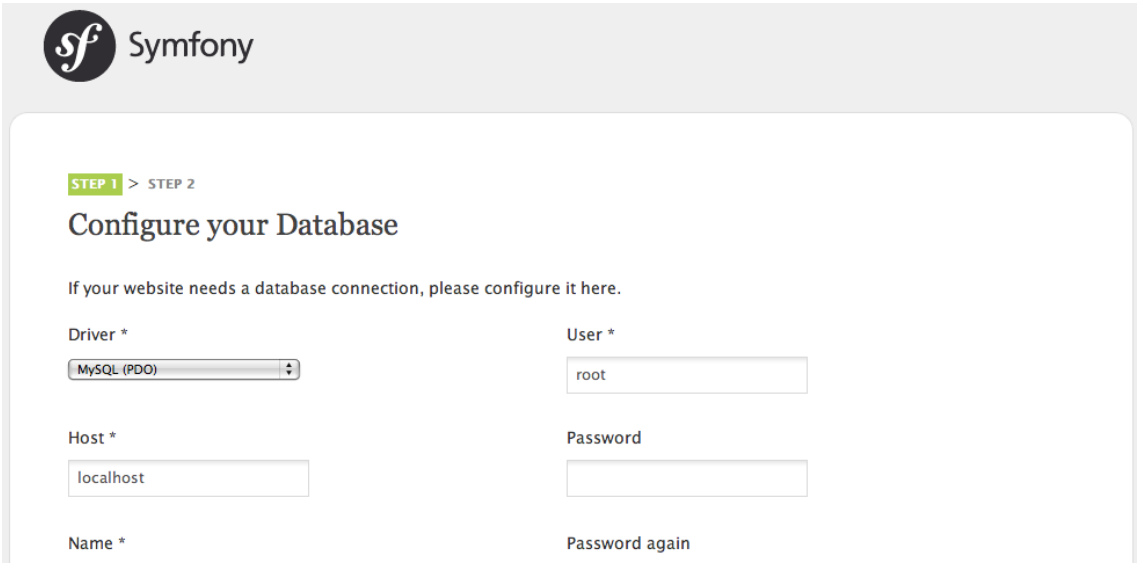


Figura 114: Pàgina d'inici de Symony2

La fallida més comuna en aquest punt, es la definició nul·la de la regió i zona horària, cosa que comporta que Symfony2 no processi correctament els components necessaris per a la seva execució, amb la qual cosa haurem de modificar l'arxiu de configuració php.ini del servidor Apache. Introduirem les dos instruccions següents:

- Php_flag short_open_tagg off
- Php_value date.timezone "Europe/Madrid"

El següent és configurar l'aplicació mitjançant l'ajuda del propi configurador de Symfony2. Per a la realització d'aquesta acció, emprarem la URL http://localhost/Symfony_Paradeta/web/config.php. Accedirem al primer enllaç (*Configure your Symfony Application online*).



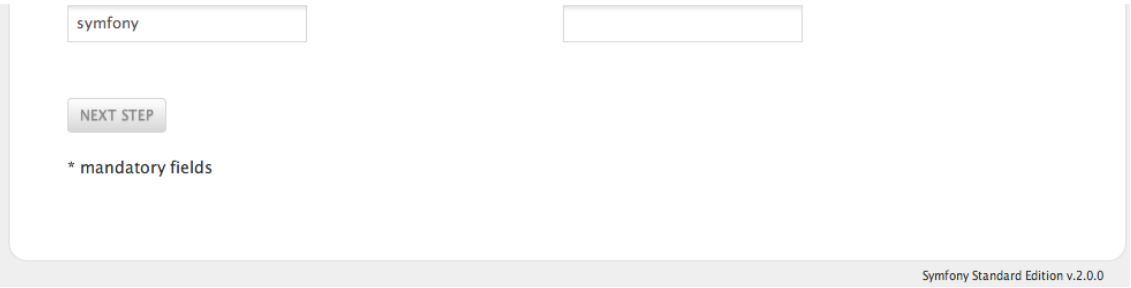


Figura 115: Configuració automàtica de symony2

La plana web presentada en la figura 115 permet configurar la base de dades sobre la qual treballarà l'aplicació. Emplenat el formulari, premerem el botó *Nest Step*.

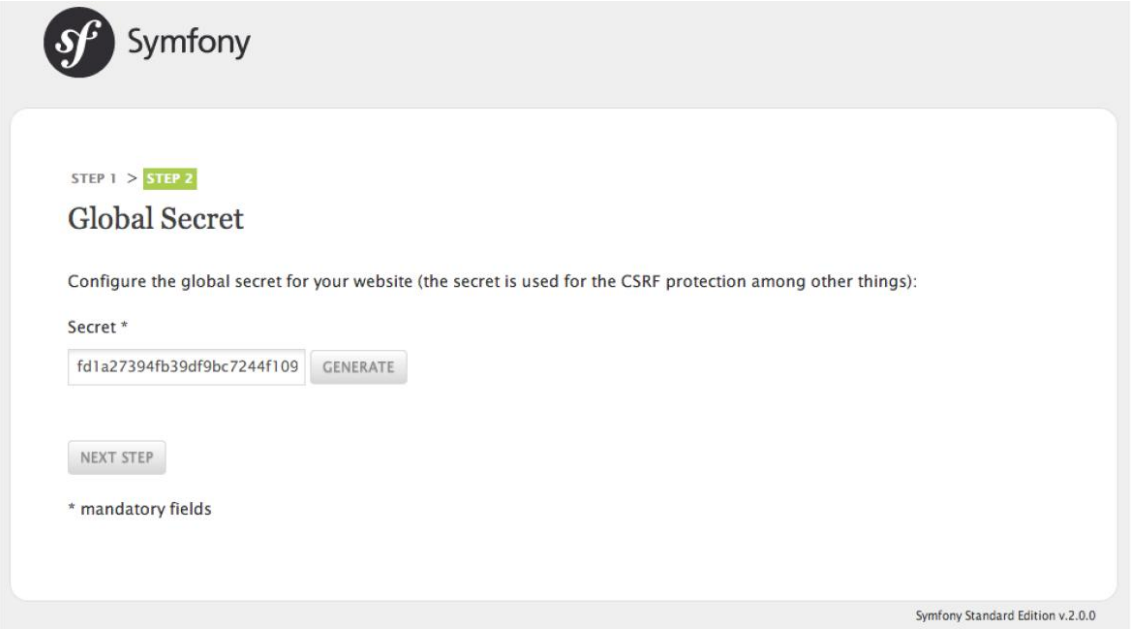


Figura 116: Generació del Secret Global

La següent plana web ens permetrà configurar el "*Secret Global*", el qual és una cadena de text aleatòria emprada en diverses parts de l'aplicació, com per exemple la protecció dels formularis davant d'atacs de tipus CSRF.

Configuració de permisos

La principal causa d'errors a l'execució de l'aplicació o els scripts de Symfony2, és la configuració nul·la de permisos sobre els directoris app/cache/ i app/logs/, els quals Symfony2 realitza el procediment d'escriptura. Com a conseqüència, executarem la següent acció:

- Chmod –R 777 app/cache app/logs.

Scripts bàsics emprats al llarg del projecte

Duran el transcurs de l'elaboració del projecte, s'han executat les següents sentencies, amb la finalitat de resoldre els conflictes generats, que comprenen des de la falta de permisos, fins a la copia de directoris:

- php vendor/bundles/Sensio/Bundle/DistributionBundle/Resources/bin/build_bootstrap.php, el qual crea un arxiu PHP amb el codi de totes les classes de l'aplicació. Es tracta d'un arxiu el qual accelera l'execució de Symfony2 i només s'ha d'executar cada cop que s'actualitza el framework.
- php app/console assets: install web, el qual copia al directori públic del projecte els arxius CSS, JavaScript e imatges).
- php app/console cache:clear – env=dev i php app/console cache:clear – env=prod, el qual borra la *cache* de desenvolupament i producció de l'aplicació.

Annex B: Configuració de la base de dades

Usuaris i permisos

Mitjançant el gestor de base de dades MySQL, crearem la taula associada a l'aplicació:

```
$ mysql -u root

mysql> CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';
```

Figura 117: Configuració d'accés a la base de dades

Els valors usuario i contraseña identificaran les credencials de l'usuari associat a la base de dades.

Acte seguit de la creació de l'usuari, l'atorgarem els permisos adequats sobre la base de dades de l'aplicació. No cal crear prèviament la base de dades, però si s'ha d'escriure el seu nom:

- GRANT ALL ON paradeta.* TO 'paradeta'@'localhost';

Configuració de l'accés a la base de dades

L'única informació requerida per symfony2 per accedir a la base de dades són les credencials de l'usuari i el nom de la base de dades. Aquesta informació és va indicar prèviament durant el transcurs del procés d'instal·lació de Symfony2. La modificació manual dels valors introduïts es realitzarà a l'arxiu app/config/parameters.ini.

Realment, l'arxiu parameters.ini no es la base d'obtenció de les dades de Doctrine2. L'accés a la base de dades es configura a l'arxiu app/config/config.yml, sota la clau dbal de doctrine.

```yaml
# app/config/config.yml
doctrine:
    dbal:
        driver:   %database_driver%
        host:     %database_host%
        port:     %database_port%
        dbname:   %database_name%
        user:     %database_user%
```

```
password: %database_password%
charset:  UTF8
```

Figura 118: Arxiu config.yml

Les opcions el nom de les quals es troba agregat per %, prenen el seu valor de les opcions amb el seu equivalent definides a l'arxiu parameters.ini.

Creació de la base de dades i taules

Configurat l'accés a les dades, crearem la base de dades buida mitjançant la llibreria que symofny2 posa a la nostra disposició.

```
$ php app/console doctrine:database:create

Created database for connection
```

Figura 119: Sentencia de creació de la base de dades

A continuació, crearem tota l'estructura de les taules de la base de dades (també anomenada com *exquema*).

```
$ php app/console doctrine:schema:create
```

Figura 120: Sentencia de creació de les taules

Si en comptes de crear les taules, solament volem observar les sentencies SQL que aquest executa, concatenarem l'opció --dumpl-sql al final de la sentencia.

Duran el transcurs del desenvolupament de l'aplicació, sol ser habitual afegir o eliminar propietats de les entitats. En aquest cas no és necessaris eliminar la base de dades i tornar-la a crear, tant sols és necessari emprar la sentencia doctrine:schema:update.

```
// Ver las sentencias SQL que se ejecutarían para la actualización
$ php app/console doctrine:schema:update --dump-sql

// Ejecutar las sentencias SQL anteriores
$ php app/console doctrine:schema:update --force
```

Figura 121: Sentencia d'actualització de les taules

Bibliografia i netgrafia

Netgrafia

- MySQL
  - Documentació MySQL 5:

    http://dev.mysql.com/doc/refman/5.5/en/tutorial.html
  - Documentació eina phpMyAdmin:

    http://www.phpmyadmin.net/home_page/docs.php/
- Apache
  - Documentació Apache:

    http://wiki.apache.org/httpd/
- PHP
  - Manual PHP:

    http://www.php.net/manual/en/

    http://es.php.net/pdo
- CSS
  - Manual CSS:

    http://www.css3.info/

    http://www.w3schools.com
- Google Maps
  - Documentació Google Maps:

    https://developers.google.com/maps/documentation/javascript/?hl=es-CL

    http://www.maestrosdelweb.com/editorial/google-maps-api-v3-introduccion-y-primeros-pasos/

- Symfony2:
  - Documentació del framework:

    http://es.wikipedia.org/wiki/Framework

    http://es.wikipedia.org/wiki/Symfony

    http://symfony.com/doc/current/index.html

    http://gitnacho.github.com/symfony-docs-es/
  - Documentació de Twig:

    http://twig.sensiolabs.org/documentation
  - Documentació Doctrine2:

    http://doctrine-orm.readthedocs.org/en/2.0.x/

    http://yaml.org/

Bibliografia

Desarollo web ágil con Symfony2, Javier Eguiluz, Primera edición, Any 2012.

Practical symfony 1.4 for Doctrine, Fabien Potencier, ISBN: 9782918390169

Jobeet Tutorial v1.3, Fabien Potencier, Any 2008

PHP Cookbook 2nd Edition, Adam Trachtenberg, August 25 of 2006, ISBN: 9780596101 015